

Sonderheft Nr. 89

Preis 14 DM

105 öS, 14 sfr.



DAS APPLE- SONDERHEFT

DOS-Organisation

ROM-Routinen

Jede Menge Software!



Sie haben einen Apple...



**wir haben die Software
und die Hardware...
wir haben die Bücher
und die Zeitschriften*...**

pandasoft

ALLES FÜR DEN APPLE II, II+, IIE
HARDWARE · SOFTWARE · BÜCHER · ZEITSCHRIFTEN

***Fordern Sie unseren
Gratiskatalog an!**

UNSERE ADRESSE:

pandasoft

UHLANDSTR. 195 · D-1000 BERLIN 12
TEL. (030) 310 423

Ich besitze einen Apple. Bitte schicken Sie mir Ihren
kostenlosen Katalog.

Name:

Adresse:

Vorwort

Wenn im Titel dieses Sonderhefts abkürzend vom „Apple“ die Rede ist, dann ist damit eigentlich der Apple-II gemeint – nicht nur einer der ältesten, sondern gleichzeitig auch einer der langlebigsten und am häufigsten von Nachbauern kopierten Computer. Sicher gibt es inzwischen Geräte, die mit moderneren Bauelementen in höher integrierter Technik aufgebaut sind als der Ur-Apple, so zum Beispiel die kompatiblen Geräte II-e und II-c desselben Herstellers oder die Computer von Blaupunkt, Basis und den zahlreichen meist taiwanesischen Kopisten. Doch setzte Apple damit einen Standard, wie man ihn höchstens noch mit dem IBM-PC vergleichen kann.

Warum ist der Apple-II so langlebig? Vermutlich liegt das an seinem flexiblen Konzept, sprich, an den „Slots“, in die man Steckkarten für individuelle Anwendungen und Aufgaben einsetzen kann; darunter auch andere Prozessoren als den 6502, zum Beispiel die CPU Z80 für das Betriebssystem CP/M oder den 16-Bit-Prozessor 68000. Daher ist es für den Apple-Besitzer möglich, mit der fortschreitenden technischen Entwicklung Schritt zu halten, ohne sich dafür regelmäßig einen neuen Computer kaufen zu müssen. (Leider hat Apple beim II-c das Slot-Konzept fallengelassen.)

mc hat seit ihrem Bestehen regelmäßig Beiträge über den Apple-II veröffentlicht. Viele der früheren Hefte sind allerdings längst nicht mehr lieferbar, obwohl die dort abgedruckten Programme und Hardware-Vorschläge nichts von ihrer Aktualität verloren haben. So reifte der Entschluß, den zahlreichen Apple-Benutzern endlich ein Kompendium von älteren und neueren Beiträgen zu ihrem Computer zu bieten, das nicht nur viele Informationen enthält, die Apple in seinen Handbüchern gewissenhaft verschweigt, sondern auch manche Problemlösung zeigt, die mit keinen oder nur geringen Anpassungen für den individuellen Anwender direkt einsetzbar ist.

Zu manchen Beiträgen gab es spätere Nachträge, Ergänzungen und Leserbriefe. Auch sie sind in diesem Sonderheft abgedruckt, da sie oftmals wertvolle Zusatzinformationen enthalten und z. B. auf nötige Software-Änderungen bei abweichenden Gerätekonfigurationen eingehen.

So. Jetzt dürften Sie wohl erst mal für eine Weile beschäftigt sein...

*Ihre
Redaktion*

Inhalt

Vorwort	1
---------------	---

Grundlagen

Apple-DOS: Arbeitsweise und Aufbau	3
Vom Umgang mit	
Apple-Maschinenprogrammen	34
ROM-Routinen des	
Applesoft-Basic-Interpreters	44
Ein Blick in Apple-DOS 3.3	58

Hardware

Tastaturpuffer für den Apple-II	27
Logikanalysator als Apple-Zusatzeinheit	61
Apple lernt Kleinschrift	75
Mehr Leistung für Apple-II	76
Apple-II: Diskettenkapazität preiswert erhöht ..	77
Universal-Schnittstelle für Apple-II	78

Software

Apple macht Textdatei aus Speicherbereich ...	10
Apple-Grafik füllt eine DIN-A4-Seite	11, 43
Apple-TED – ein komfortabler Texteditor	13
Datenaustausch	17
Felder schnell abgespeichert	20
Mehr Platz auf Apple-Disketten	23
Programmtext-Editor	24
Prüfsummen-Programm für den Apple-II	26
Relocator für den Apple-II	28
Spooler beschleunigt Druckausgabe	29
Apple liest CBM-Dateien	33

Apple-II auf Literatursuche	35
Apple-II lernt sprechen	38
Apple-II liest und druckt Strichcode	41
Shapemaker spart mühsame Kleinarbeit	51
Strichcode drucken und lesen	53
Apple-II liest Strichcode	55
Apple-II steuert Fernschreiber	59
Apple-II sucht Bytes	60
Autostart und Programmschutz	
für Apple-II-plus	62
DOS-Umschaltung beim Apple	64
MX-80 druckt Apple-Grafik	67
Präzises Paginieren per „PUT“	70
V.24-Ein-/Ausgabe für den Apple	74
Kommunikation mit dem Apple-II	86
Bytefolgen schnell gefunden	89
Apple-Disk-Editor	91

Tips und Tricks

Apple-Kniffe	69
Groß- und Kleinschreibung ohne Shift-Taste ...	19
Catalog-Stopp	25
Nervenschonender Cursor für den Apple	32
Apfel-Menü	32
ASCII-Zeichenfolgen sichtbar gemacht	37
Centronics – ganz einfach	40
Grafik mit dem MX-80	50
Step und Trace für Apple-II+ und Apple-IIe ...	63
Apple-Eigenheiten	66
Listbare Autostart-Programme	66
Nachträge	95
Produktanzeigen	Umschlags., 12, 40

Impressum: 1984, Franzis-Verlag GmbH, Karlstraße 37–41, D-8000 München 2. 3., verbesserte Auflage.

Bearbeitet von der Redaktion der Zeitschrift **mc**. Für den Text verantwortlich: Dipl.-Ing. (FH) Herwig Feichtinger.

© Sämtliche Rechte – besonders das Übersetzungsrecht – an Text und Bildern vorbehalten. Fotomechanische Vervielfältigung nur mit Genehmigung des Verlages. Jeder Nachdruck, auch auszugsweise, und jede Wiedergabe der Abbildungen, auch in verändertem Zustand, sind verboten.

ISSN 0722-0022. Druck: Franzis-Druck GmbH, München. Printed in Germany, Imprimé en Allemagne. ZV-Art.-Nr. 89041 · F/ZV/1284/1086/5'

Dr. Ralf Wiegandt

Apple-DOS: Arbeitsweise und Aufbau

Die überwiegende Mehrheit der Apple-II-Besitzer verwendet Disketten zur Abspeicherung von Daten und Programmen und ist daher mit dem Disketten-Betriebssystem DOS (Disk Operating System) vertraut. Eine detaillierte Kenntnis der Vorgänge bei der Diskettenverwaltung kann sich als sehr hilfreich bei der Erstellung von Maschinenprogrammen unter Benutzung von DOS-Systemroutinen, bei der „Reparatur“ von unlesbar gewordenen Disketten und bei der individuellen Anpassung des DOS an eigene Wünsche erweisen.

Geschichte des Disketten-Betriebssystems

Die Apple Computer Inc. stellte im Jahre 1978 mit der Einführung des Disk-II-Laufwerks das „Disk Operating System“ (Version 3.1) zur Verfügung. Zu diesem Zeitpunkt noch unausgereift, enthielt das Betriebssystem eine ganze Anzahl von Fehlern und war nur spärlich dokumentiert. Im Zuge der Entwicklung des Apple-II-Plus und des Autostart-ROM wurde daher bereits im folgenden Jahr eine neue Version vorgestellt. Obwohl sich DOS 3.2 in sehr vielen Einzelheiten von seinem Vorgänger unterschied, blieb die grundlegende Struktur weitgehend erhalten. Inzwischen war jedoch bereits ein 178 Seiten starkes Handbuch verfügbar, welches den Gebrauch der DOS-Kommandos ausführlich beschrieb und auch einige Einblicke in die interne Arbeitsweise vermittelte. DOS 3.1 und DOS 3.2 verwendeten das gleiche Aufzeichnungsformat mit einer maximalen Speicherkapazität von 113,75 KByte pro Diskette. Durch eine geschicktere Methode der Abspeicherung konnte die Kapazität einer Diskette mit der Einführung der DOS-Version 3.3 im August 1980 um fast 25 % auf 140 KByte erhöht werden. Damit entspricht die Strukturierung des Disketteninhaltes unter der bisher letzten DOS-Version auch dem Format der anderen für Apple verfügbaren Sprachen und Betriebssysteme (Pascal, Fortran, CP/M). Selbstverständlich stellte Apple mit dem neuen

System auch die entsprechende Software zur Verfügung, die es erlaubt, Disketten des alten Formats weiterhin zu lesen bzw. in das neue Format zu verwandeln.

Datenspeicherung auf Diskette

Zur Strukturierung des Disketteninhaltes wird eine Diskette während der Initialisierung in Spuren und Sektoren eingeteilt. Spuren sind konzentrische Ringe auf der Oberfläche der Diskette, die durch ihren Abstand vom Mittelpunkt gekennzeichnet sind. Ähnlich wie der Arm eines Plattenspielers kann der Schreib-/Lesekopf des Diskettenlaufwerks über einer bestimmten Spur positioniert werden. Prinzipiell kann eine

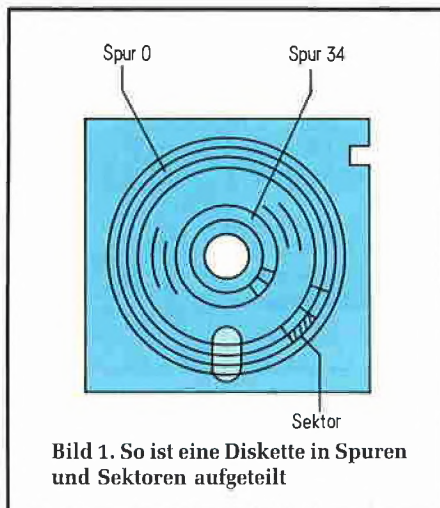


Bild 1. So ist eine Diskette in Spuren und Sektoren aufgeteilt

einfache 5-Zoll-Diskette maximal in etwa 70 Spuren unterteilt werden. Apple-DOS faßt aus technischen Gründen jeweils zwei mögliche Spuren zusammen und formatiert eine Diskette in 35 konzentrischen Ringen, die die Nummern 0 bis 34 tragen. Dabei bezeichnet 0 die äußerste Spur und 34 die innerste (bei Disketten, die einen besonderen Kopierschutz tragen, wird gelegentlich auch eine 36. Spur verwendet). Bild 1 zeigt die Lage der (für das Auge unsichtbaren) Spuren auf einer Diskette.

Jede Spur ist in einzelne Sektoren aufgeteilt. Ein Sektor ist die kleinste Einheit, die mit einem Mal gelesen, geschrieben oder verändert werden kann, er enthält genau 256 nutzbare Datenbytes. Apple unterscheidet zwei verschiedene Formate: Unter DOS 3.1 und 3.2 wird eine Spur in 13 Sektoren (numeriert von 0 bis 12), unter dem neuen DOS 3.3 in 16 Sektoren (0 bis 15) unterteilt. Eine DOS-Besonderheit ist die Tatsache, daß das auf den Disketten enthaltene „Indexloch“ nicht verwendet wird, um den ersten Sektor einer Spur zu finden. Statt dessen wird ein Verfahren angewandt, welches es der Software erlaubt, ohne Hilfe der Hardware jede Spur und jeden Sektor aufzufinden. Dieses Verfahren (Soft Sectoring) ist zwar platz- und zeitaufwendiger bei der Abspeicherung, erlaubt jedoch einen hohen Grad an Flexibilität.

Soft-Sektorierung

Zur Verwaltung der Diskettenaufzeichnung ist es notwendig, neben den Daten des Benutzers (256 Byte/Sektor) eine ganze Anzahl von Informations- und Steuerdaten auf der Diskette abzuspeichern. Hierzu wird eine Spur in einzelne Felder aufgeteilt, die eine unterschiedliche Anzahl von Bytes enthalten können. Es wird unterschieden zwischen den „Adreßfeldern“ mit Informationen über Spur- und Sektornummer und den „Datenfeldern“, die im wesentlichen die Benutzerdaten in einer besonderen, verschlüsselten Form enthalten. Zwei aufeinanderfolgende Felder sind jeweils durch eine Lücke getrennt, die aus einer Anzahl besonderer „Synchronisationsbytes“ besteht. Die Anordnung der Adreß- und Datenfelder auf einer Spur ist in Bild 2 dargestellt. Den Synchronisationsbytes in den Lücken zwischen den Datenfeldern kommt eine ganz besondere Bedeutung bei der Methode des „Soft-Sectoring“ zu. Anders als gewöhnliche Datenbytes bestehen sie nicht aus acht, sondern aus zehn einzelnen Bits, von denen die ersten

acht den Wert 1 und die letzten beiden den Wert 0 haben. Durch diese besondere Struktur sind sie in der Lage, die Hardware mit den Datenbytes auf der Diskette zu synchronisieren. Physikalisch ist eine Spur ja nichts anderes als ein kontinuierlicher Strom von einzelnen Bits, und es gibt in der Tat zunächst keine Möglichkeit, festzustellen, wo ein Byte beginnt oder endet. Wenn das Laufwerk den Befehl erhält, Daten zu lesen, wird es irgendwo innerhalb der ca. 50000 Bits auf einer Spur damit beginnen. Es würden jedoch nur dann die korrekten Daten übermittelt, wenn der Startpunkt zufällig mit dem Beginn eines Datenbytes identisch ist (die Wahrscheinlichkeit dafür ist aber gering). Diese Schwierigkeit wurde bei Apple auf folgende einfache Art gelöst: Zunächst wird die Hardware so ausgelegt, daß sie als erstes Bit eines Bytes stets die 1 erwartet. Werden zunächst Nullbits angetroffen, so werden sie so lange überlesen, bis eine 1 auftaucht (diese Konvention beschränkt natürlich die Anzahl der möglichen Datenbytes auf 128!). Damit werden die erwähnten Synchronisationsbytes ungeachtet ihrer beiden zusätzlichen Nullen als Hexadezimalwert FF gelesen. Durch eine Folge mehrerer solcher Werte lassen sich die Lücken zwischen Adreß- und Datenfeldern eindeutig erkennen.

Die zwei zusätzlichen Bits bewirken, daß der Lesekopf (unabhängig von seinem „Aufsetzpunkt“) nach höchstens fünf Synchronisationsbytes immer genau am Anfang eines neuen Bytes steht, so daß alle folgenden Daten korrekt interpretiert werden.

Daß dieses Verfahren tatsächlich in der beschriebenen Weise funktioniert, macht die Betrachtung der folgenden Bitsequenz deutlich, die fünf Synchronisationsbytes darstellt:

```
1111111100111111110011111
1110011111111001111111100
```

Unter Berücksichtigung der Tatsache, daß das erste Bit eines gelesenen Bytes den Wert 1 haben muß, ergeben sich hier – je nach dem Einsetzpunkt des Lesevorgangs – die folgenden Byte-Inhalte (hexadezimal):

- a) Start beim ersten Bit
FF FF FF FF FF
- b) Start beim zweiten Bit
FE FF FF FF FF
- c) Start beim dritten Bit
FC FF FF FF FF
- d) Start beim vierten Bit
F9 FE FF FF FF
- e) Start beim fünften Bit
F3 FC FF FF FF

- f) Start beim sechsten Bit
E7 F9 FE FF FF
- g) Start beim siebenten Bit
CF F3 FC FF FF
- h) Start beim achten Bit
9F E7 F9 FE FF

Das neunte und zehnte Bit werden überlesen, da sie nicht die Bedingung (Anfangsbit = 1) erfüllen, womit der Zyklus beendet ist. Man erkennt, daß unabhängig vom Startpunkt spätestens nach fünf Synchronisationsbytes eine korrekte Synchronisation erreicht ist. Die Größe der Lücken zwischen den einzelnen Feldern ist von Spur zu Spur verschieden und auch abhängig vom verwendeten Laufwerk. DOS erzeugt zunächst große Lücken, welche dann so lange verkleinert werden, bis eine ganze Spur geschrieben werden kann, ohne sich selbst zu überlappen. Dabei ist jedoch wichtig, daß mindestens fünf Synchronisationsbytes in jeder Lücke übrigbleiben.

Das Adreßfeld

Der Aufbau eines Adreßfeldes ist in Bild 3 dargestellt. Es enthält Informationen über das nachfolgende Datenfeld, (eingeraht von einem Vorspann Header), bestehend aus den Bytes D5, AA, 96, und einem Nachspann (Trailer) mit den Bytes DE, AA, E5. Die Verschlüsselung der Datenbytes geschieht so, daß die Bytes D5 und AA außer im Header eines Feldes nirgendwo vorkommen können – sie sind reserviert. Am Auftreten dieser Bytefolge erkennt DOS also eindeutig den Beginn eines Adreß- oder Datenfeldes.

Die eigentlichen Daten des Adreßfeldes werden abgeschlossen durch eine Prüfsumme, die aus den vorangegangenen Daten gebildet wird und zur Überprüfung der Richtigkeit der gelesenen Daten herangezogen wird.

Das Datenfeld

Das Datenfeld besteht, wie das Adreßfeld, aus einem Header, den Daten, einer Prüfsumme und einem Trailer (Bild 4). Der Header unterscheidet sich nur durch das letzte Byte (AD) von dem des Adreßfeldes.

Wie bereits erwähnt, ist die Apple-Hardware so ausgelegt, daß nicht alle 256 möglichen Bytes von der Diskette gelesen werden können. Dies bedeutet, daß die Benutzerbytes codiert werden müssen, wozu DOS 3.3 zwei verschiedene Verfahren verwendet. Das erste wird im Adreßfeld benutzt und besteht darin, ein Datenbyte in zwei Diskettenbytes zu zerlegen, von denen eines die ungeraden und das andere die geraden Bits enthält. Ein Datenbyte

DB = ABCDEFGH

wird also zerlegt in die beiden Bytes:

B1 = 1A1C1E1G

B2 = 1B1D1F1H.

Man könnte somit eine eindeutige Umrechnungstabelle angeben, die folgendermaßen aussähe:

00 → AA AA	FC → FE FE
01 → AA AB	FD → FE FF
02 → AB AA	FE → FF FE
03 → AB AB	FF → FF FF

Der Nachteil dieser Art der Verschlüsselung besteht darin, daß man zur Abspeicherung von 256 Datenbytes 512 Diskettenbytes benötigt, so daß eine Spur höchstens zehn Datensektoren aufnehmen könnte. Das ergäbe eine Speicherkapazität von etwa 88 KByte/Diskette, also etwa 60 % des unter DOS 3.3 tatsächlich verfügbaren Speicherplatzes. Zur Verschlüsselung der Bytes im eigentlichen Datenfeld wird deshalb eine andere Methode angewandt, die als „6-zu-2-Codierung“ bezeichnet wird (ältere DOS-Versionen benutzten ein anderes Verfahren). Der Methode liegen drei For-

Bild 2. Aufbau einer Diskettenspur

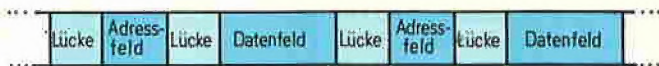


Bild 3. So ist ein Adreßfeld aufgebaut

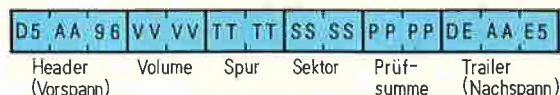
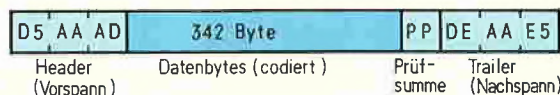


Bild 4. So ist ein Datenfeld aufgebaut



derungen an die Hardware zugrunde:
1. Das erste Bit eines gültigen Datenbytes muß den Wert 1 haben. 2. Das Byte darf maximal ein Paar von Nullbits enthalten. 3. Es müssen stets mindestens zwei benachbarte Bits auf 1 gesetzt sein (ausgenommen das höchstwertige Bit). Nimmt man die reservierten Bytes AA und D5 heraus, so werden diese Bedingungen genau von 64 Bytes zwischen 96 und FF erfüllt.

Verschlüsselung

Der Codierungsvorgang geht nun folgendermaßen vor sich: Zunächst werden die 256 Bytes eines Sektors nach einem komplizierten Muster in 6-Bit-Bytes (mit führenden Nullen) verwandelt und in zwei getrennten Puffern („Primär- und Sekundärdatenpuffer“) abgelegt. Bei diesem auch als „Prenibbilizing“ bezeichneten Verfahren entstehen aus den 256 Benutzerbytes genau 342 Diskettenbytes. Diese werden anschließend der Reihe nach paarweise durch EOR (Exklusiv-ODER) verknüpft und gemäß Tabelle 1 für den „6-zu-2-Code“ in die endgültig abgespeicherten Bytes umgewandelt.

Dieses Verschlüsselungsverfahren scheint auf den ersten Blick unnötig kompliziert. Es ist jedoch so gewählt, daß einerseits ein eindeutiges „Soft-Sectoring“ möglich wird und andererseits alle Schreib- und Leserzriffe in kürzester Zeit durchgeführt werden können. Dies ist der Fall, weil während des Lesens lediglich EOR-Verknüpfungen zwischen den Bytepaaren durchgeführt werden müssen, um die korrekten Daten zu erhalten. Nebenbei fällt dabei nach dem letzten verarbeiteten Byte auch die Prüfsumme ab, die zur Überprüfung des korrekten Lesevorganges verwendet wird.

Zur Verringerung der Zugriffszeit wendet DOS noch einen weiteren Trick an: Das „Sector-Interleaving“. Normalerweise entsteht zwischen dem Lesen oder Schreiben zweier aufeinanderfolgender Sektoren eine kleine Zeitverzögerung, während der sich die Diskette bereits ein Stück weitergedreht hat. Es ist daher sinnvoll, die logische Reihenfolge der Sektoren auf der Diskette zu verändern, um Wartezeiten zu verringern. Während die physikalischen Sektoren der Reihe nach von 0 bis F durchnummeriert werden, werden die logischen Sektoren in der Reihenfolge 0,D,B,9,7,5,3,1,E,C,A,8,6,4,2,F abgelegt. DOS behandelt also z. B. den physikalischen Sektor 2 bei allen Lese- und Schreibvorgängen als logischen Sektor B, den physikalischen Sektor D als logischen Sektor 4 usw.

Tabelle 1: Die 256 Bytes für einen Sektor werden zuerst in 342 Bytes mit je zwei führenden Nullen aufgespaltet, die nach dem dargestellten Schema umgesetzt und abgespeichert werden

00 = 96	10 = B4	20 = D6	30 = ED
01 = 97	11 = B5	21 = D7	31 = EE
02 = 9A	12 = B6	22 = D9	32 = EF
03 = 9B	13 = B7	23 = DA	33 = F2
04 = 9D	14 = B9	24 = DB	34 = F3
05 = 9E	15 = BA	25 = DC	35 = F4
06 = 9F	16 = BB	26 = DD	36 = F5
07 = A6	17 = BC	27 = DE	37 = F6
08 = A7	18 = BD	28 = DF	38 = F7
09 = AB	19 = BE	29 = E5	39 = F9
0A = AC	1A = BF	2A = E6	3A = FA
0B = AD	1B = CB	2B = E7	3B = FB
0C = AE	1C = CD	2C = E9	3C = FC
0D = AF	1D = CE	2D = EA	3D = FD
0E = B2	1E = CF	2E = EB	3E = FE
0F = B3	1F = D3	2F = EC	3F = FF

Das „Sector-Interleaving“ wird auch auf Pascal-, Fortran- und CP/M-Disketten verwendet; allerdings ist dort die Reihenfolge der logischen Sektornummern eine andere als unter DOS 3.3.

Dateien-Format

Eine DOS-3.3-Diskette enthält drei verschiedene Arten von Information. Zunächst einmal ist das vollständige Betriebssystem im Maschinencode abgespeichert, so daß im Prinzip jede beliebige Diskette beim Einschalten des Apple als „Boot-Diskette“ zum Laden des Betriebssystems verwendet werden kann. Das DOS-Image verbraucht 37 Sektoren und ist auf den Spuren 0, 1 und 2 abgelegt (Die Sektoren 5 bis 15 der Spur 2 werden zwar nicht benötigt, sind aber trotzdem reserviert und können in der Regel nicht vom Benutzer verwendet werden).

Die zweite Gruppe von Informationen befindet sich auf der Spur Nr. 17 und besteht aus einem Inhaltsverzeichnis und einer Belegungstabelle der Diskettensektoren. Diese Tabelle, die als „Volume Table of Contents“ (VTOC) bezeichnet wird, enthält für jeden Diskettensektor die Information, ob dieser bereits belegt oder noch frei ist. Vor jeder Schreibaktion zieht DOS das VTOC heran, um den nächsten freien Sektor zu finden. Der genaue Aufbau des VTOC ist im DOS-Handbuch ausführlich beschrieben und soll hier nicht noch einmal ausgeführt werden (Bild 5).

Die eigentlichen, vom Benutzer verwendeten Daten sind auf der Diskette in Form von Dateien abgespeichert. Hierzu verbleiben von den 35 Spuren nach Abzug der Spuren 0, 1, 2 und 17 noch 31 Spuren, entsprechend einem verfügbaren Speicherplatz von 124 KByte. DOS unterscheidet acht verschiedene Dateitypen, von denen jedoch augenblicklich nur fünf verwendet werden. Diese fünf Typen werden folgendermaßen gekennzeichnet:

- Typ T: Textdatei (Text oder Daten im ASCII-Format)
- Typ B: Binärdatei (Daten oder Maschinenprogramme im Binärformat)
- Typ A: Applesoft-Datei (Applesoft-Programme)
- Typ I: Integer-Datei (Integer-Basic-Programme)
- Typ R: Reloizierbare (verschiebbare) Datei (Maschinenprogramme; bisher nur vom 'DOS-Toolkit' verwendet).

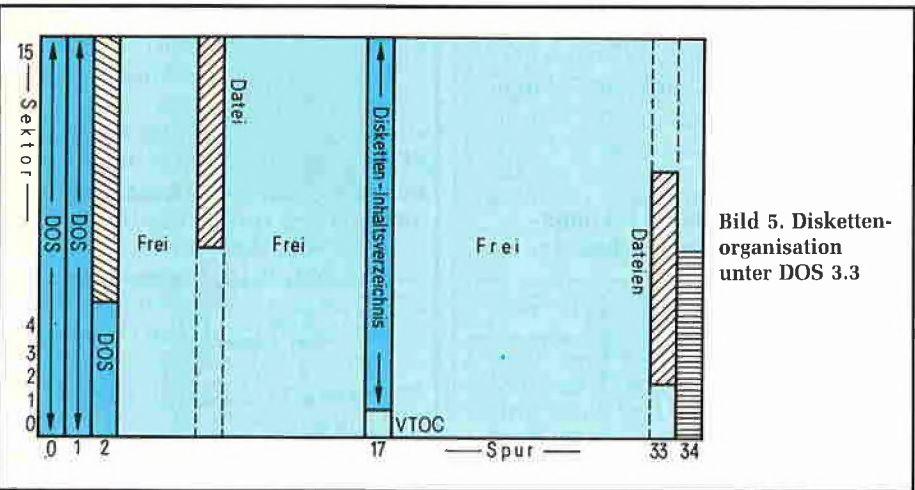


Bild 5. Diskettenorganisation unter DOS 3.3

Jede Datei besteht aus einem oder mehreren Datensektoren, einem Eintrag im Disketteninhaltsverzeichnis und einer sogenannten „Spur-/Sektor-Liste“ oder „Track/Sektor List“ (TSL). Diese Liste ist notwendig, da unter DOS 3.3 die Dateisektoren nicht zusammenhängend hintereinander abgespeichert werden (wie etwa in UCSD-Pascal), sondern – je nach verfügbarem Speicherplatz – überall auf der Diskette verstreut sein können.

Die Sektor-Belegungs-Liste

Selbstverständlich werden auf einer neuen, ungebrauchten Diskette Dateisektoren zunächst hintereinander abgelegt; durch wiederholtes Löschen und Neuabspeichern von Dateien entstehen jedoch an verschiedenen Stellen unterschiedlich große Lücken, so daß nach längerem Gebrauch eine zusammenhängende Abspeicherung nicht mehr möglich ist. Damit DOS dennoch die Dateisektoren in der richtigen Reihenfolge auffinden kann, wird zu jeder Datei eine Liste angelegt, aus der die Position der Sektoren auf der Diskette hervorgeht. Der genaue Aufbau dieser TSL ist ebenfalls im DOS-Handbuch erläutert.

Das Disketteninhaltsverzeichnis, das die Sektoren 1 bis 15 auf Spur 17 belegt, enthält zu jeder abgespeicherten Datei einen Eintrag, welcher aus dem Dateinamen, dem Dateityp, der Dateilänge und einem Hinweis auf die Position der Spur-/Sektor-Liste besteht. Das Verzeichnis kann pro Sektor sieben Einträge, insgesamt also maximal 105 Dateieinträge aufnehmen. Es ist demnach nicht möglich, mehr als 105 Dateien auf einer DOS-Diskette abzuspeichern. Aufgrund dieser Diskettenorganisation besteht ein Dateizugriff aus mehreren Stufen. Zunächst wird im Inhaltsverzeichnis nach dem entsprechenden Eintrag gesucht und dort die Position der TSL bestimmt. Dann wird diese Liste eingelesen und aus ihr die Position der Datensektoren ermittelt. Erst im dritten Schritt kann dann auf diese Sektoren zugegriffen werden. Zusätzlich muß laufend das VTOC aktualisiert werden. Das Verfahren hat den Vorteil einer optimalen Platzausnutzung auf der Diskette, ist aber auf der anderen Seite auch recht zeitaufwendig.

Das Betriebssystem

Wenn der Apple eingeschaltet wird, ist der RAM-Speicher zunächst leer. Der Autostart-Monitor im ROM sorgt jedoch

dafür, daß das auf der eingelegten Diskette abgespeicherte DOS-Image automatisch in den Speicher geladen wird.

Dieser Vorgang wird allgemein als „Booting“ bezeichnet und läuft in mehreren Stufen ab.

Als erstes wird ein kurzes Maschinenprogramm, welches sich auf dem ROM der Controller-Karte (ab Speicheradresse \$C600) befindet, aufgerufen. Dieses zieht den Schreib-/Lesekopf zurück zur Spur 0 und liest Sektor 0 in den RAM-Speicher ab Adresse \$800. Dieser Sektor enthält wiederum ein kurzes Maschinenprogramm, welches in der nun folgenden Stufe aufgerufen wird. Dabei werden die nächsten neun Sektoren auf Spur 0 (Sektoren 1 bis 9) eingelesen, welche ihrerseits für die nächste Stufe des Prozesses verwendet werden. Die Adresse, an die diese neun Spuren geladen werden, hängt davon ab, ob es sich bei der Diskette um einen „Master“ oder einen „Slave“ handelt. Wird DOS von einer „Slave“-Diskette eingelesen, so wird es stets in demselben Speicherbereich abgelegt, in dem es sich beim Initialisieren der Diskette befand. Das bedeutet, daß eine auf einem 48-K-Apple initialisierte „Slave“-Diskette auch nur auf einem 48-K-Apple „gebootet“ werden kann. Dies ist anders bei einer „Master“-Diskette.

Diese enthält in den Sektoren 10 und 11 der Spur 0 einen „Relocator“, der in der Lage ist, das geladene DOS-Image so zu verschieben, daß es stets den oberen RAM-Bereich ausfüllt. Eine „Master“-Diskette kann also auf jedem beliebigen Apple „gebootet“ werden.

Im dritten Schritt werden nun mit Hilfe der inzwischen schon eingelesenen 10 Sektoren alle weiteren DOS-Sektoren von der Diskette geholt und im RAM-Speicher abgelegt. Abschließend wird gegebenenfalls noch eine Verschiebung des DOS-Image zur oberen Grenze des RAM mit Hilfe des „Relocators“ durchgeführt.

Tabelle 2: DOS-Bereiche bei unterschiedlichem Speicherausbau

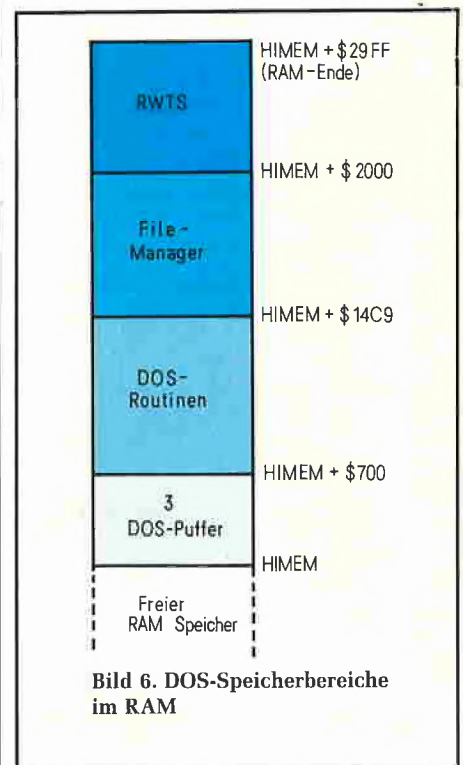
Speichergröße	32 K	36 K	40 K	48 K
HIMEM	5600	6600	7600	9600
DOS-Programme	5D00	6D00	7D00	9D00
File-Manager	6AFD	7AFD	8AFD	AAF
RWTS	77B5	87B5	97B5	B7B5
RAM-Ende	7FFF	8FFF	9FFF	BFFF

Damit ist die Prozedur des „Booting“ beendet, und DOS belegt die 8960 obersten Bytes des RAM-Speichers (8, 75 KByte).

DOS-Adressen

Der für DOS reservierte Bereich des RAM-Speichers läßt sich in vier Abschnitte einteilen (Bild 6). Die obersten 2,5 KByte bestehen aus dem RWTS-Programm (Read/Write Track/Sektor), welches das Schreiben und Lesen einzelner Diskettensektoren steuert. Die darunter befindlichen 2,8 KByte enthalten den „File-Manager“, der aus einer Anzahl von Unterprogrammen zum Bearbeiten von Dateien besteht. Die eigentlichen DOS-Routinen befinden sich in einem 3,5 KByte großen Block unterhalb des File-Managers. Schließlich reserviert DOS einen Bereich von 1,75 KByte als Dateipuffer für drei Dateien (dies ist die Voreinstellung; der Benutzer kann diesen Speicherbereich vergrößern oder verkleinern, indem er weitere Puffer hinzufügt bzw. die Zahl der Puffer verringert).

Die Untergrenze des DOS-Bereiches wird durch den Wert für „HIMEM“ (High Memory) festgelegt. Diese Zahl ist als Adresse in der „Zero-Page“ des Apple abgelegt und definiert die höchste für den Benutzer verfügbare RAM-Adresse. Der Wert von HIMEM und damit die



Adressen der verschiedenen DOS-Speicherbereiche sind von der Speichergröße des Rechners abhängig. Die möglichen Werte (hexadezimal) zeigt *Tabelle 2*. Häufig erscheint es wünschenswert, DOS-Routinen von eigenen Maschinenprogrammen aus aufzurufen. Da die Lage von DOS im Speicher jedoch von der Ausbaustufe des Rechners abhängt, müßte man dazu stets eine Tabelle der Einsprungadressen, bezogen auf die jeweilige Speichergröße, heranziehen. Glücklicherweise ist dies jedoch nicht notwendig, da DOS in Speicherseite 3 eine Gruppe von „Vektoren“ bereitstellt, die den Aufruf von DOS-Routinen über feste Adressen erlaubt. Diese Vektortabelle wird bereits beim „Booten“ des Betriebssystems aufgebaut und enthält Sprungbefehle auf die aktuellen Adressen der wichtigsten DOS-Routinen. *Tabelle 3* gibt eine Beschreibung der DOS-Vektoren.

Für die Verwendung der DOS-Routinen in eigenen Maschinenprogrammen gibt *Tabelle 4* einige wichtige Adressen an. Sie beziehen sich auf eine Ausbaustufe von 48 KByte (zur Verwendung mit einem 32-KByte-Apple ist von den Adressen hexadezimal 4000 zu subtrahieren).

Direkte Disk-Zugriffe

Die von DOS zur Verfügung gestellten Kommandos erlauben eine komfortable Behandlung von Disketten-Dateien.

Trotz seiner vielfältigen Möglichkeiten deckt das Kommandosystem nicht alle Benutzerwünsche ab. Die Eingriffe des Benutzers sind auf die Dateiebene beschränkt; ein direkter Zugriff auf einzelne Sektoren ist nicht vorgesehen. In diesem und den folgenden Abschnitten soll nun beispielhaft aufgezeigt werden, wie man unter Verwendung einfacher Programme die DOS-Routinen nutzen kann, um einzelne Diskettensektoren zu lesen oder zu verändern.

Allen vorgestellten Beispielen liegt die RWTS-Routine zugrunde, mit deren Hilfe auf den Inhalt beliebiger Sektoren in benutzerlesbarer Form (also uncodiert) zugegriffen werden kann. RWTS benötigt zwei Parameterlisten, die vom Benutzer definiert werden müssen. Es handelt sich um den „Input/Output Control Block (IOB)“ und die „Device Characteristics Table (DCT)“. Beide Listen sind im RAM-Speicher angelegt und können somit vor dem Aufruf von RWTS verändert werden. Der IOB besteht aus 17

Tabelle 3: DOS-Sprungvektoren in Speicherseite 3

Adresse	Inhalt
3D0	Sprung zur DOS-Warmstart-Routine (DOS-Initialisierung ohne Löschen des Basic-Programms im Speicher)
3D3	Sprung zur DOS-Kaltstart-Routine (DOS-Initialisierung wie nach dem „Booten“)
3D6	Sprung zum File-Manager
3D9	Sprung zur RWTS-Routine
3DC	Unterprogramm zum Aufbau der Eingabeparameter für den File-Manager
3E3	Unterprogramm zum Aufbau der Eingabeparameter für RWTS
3EA	Sprung zur DOS-Routine, in der die DOS-Anschlüsse zur Ein- und Ausgabe wiederhergestellt werden
3EF	Sprung zur Software-Interrupt-Routine
3F2	Adresse der Reset-Routine
3F4	„Power-up-Byte“ (Unterscheidung, ob Reset einen Kalt- oder Warmstart durchführt)

Tabelle 4: Wichtige DOS-Adressen für 48-KByte-Systeme (bei 32 KByte ist jeweils hex 4000 abzuziehen)

9D00	: Adresse des Dateinamen-Feldes im ersten DOS-Puffer
9D0C	: Adresse des DOS-Anfangs (= \$9D00)
9D84	: Kaltstart-Routine
9DBF	: Warmstart-Routine
9E51	: Kopie der Vektortabelle in Speicherseite 3
9EBA	: Sprung zur aktuellen Eingaberoutine
9FC5	: Sprung zur aktuellen Ausgaberoutine
A1D6	: Dezimal-Umwandlungsroutine
A203	: Hexadezimal-Umwandlungsroutine
A229	: Kommando PR#
A22E	: Kommando IN#
A233	: Kommando MON
A23D	: Kommando NOMON
A251	: Kommando MAXFILES
A316	: Kommando CLOSE (alle offenen Dateien)
A4D1	: Kommando RUN (DOS)
A4FC	: Kommando RUN (Applesoft)
A56E	: Kommando CATALOG
A57A	: Kommando FP
A59E	: Kommando INT
A884	: DOS-Kommandotabelle
A971	: DOS-Fehlertabelle
AA4F	: DOS-Variablenliste
AA75	: Puffer für Dateinamen
AAB1	: DOS-Konstanten- und Variablenliste
AAC1	: File Manager-Variable und Routinen
B600	: Boot-Routine
B7E8	: RWTS-Parameterliste (IOB)
B7FB	: DCT (Device Characteristics Table) für RWTS
B800	: „Prenibble“-Routine
B8C2	: „Postnibble“-Routine
BA29	: Übersetzungstabellen für „Pre- und Postnibbilizing“
BB00	: Primär-Datenpuffer
BC00	: Sekundär-Datenpuffer
BEAF	: Kommando INIT
BFB8	: „Sector Interleaving“-Tabelle

Die folgenden Adressen in der „Zero-Page“ werden (neben anderen) von DOS verwendet:

26, 27	: Lesepuffer-Adresse
2B	: Steckplatz * 16 für „Boot“-Laufwerk
2C-2F	: Werte aus Sektor-Header
33	: DOS-„Prompt“-Symbol
3E, 3F	: RWTS-Pufferadresse
48, 49	: RWTS-IOB-Adresse
73, 74	: HIMEM

Bytes und enthält die Parameter für den Lese- oder Schreibvorgang. Die DCT umfaßt vier Bytes mit zusätzlichen Hardware-Informationen des Disk-II-Laufwerks. Für die Benutzung mit Apple-Standardlaufwerken braucht diese Liste vom Benutzer nicht verändert zu werden.

Die Startadresse des IOB ist abhängig von der Speichergroße des Rechners (\$B7E8 bei 48 KByte, \$77E8 bei 32 KByte) und muß daher durch einen Unterprogrammaufruf über die Vektortabelle in der Speicherseite 3 (JSR \$3E3) bestimmt werden.

Vor dem RWTS-Aufruf müssen die folgenden Bytes (relativ zur IOB-Startadresse) vom Benutzer definiert werden:

Byte Nr.	Inhalt
IOB+01	Nr. des Steckplatzes*16 (\$60 für Slot 6)
IOB+02	Nr. des Laufwerkes (\$01 oder \$02)
IOB+03	Erwartete Volume-Nummer (\$00 für alle)
IOB+04	Spur (\$00...\$22)
IOB+05	Sektor (\$00...\$0F)
IOB+08	Adresse (niederwertig) des Datenpuffers
IOB+09	Adresse (höherwertig) des Datenpuffers

IOB+0C	Kommandocode: \$00 = Lesekopf nur positionieren \$01 = Sektor in Puffer einlesen \$02 = Sektor mit Pufferinhalt beschreiben \$04 = Diskette formatieren (!)
IOB+0E	wie IOB+03
IOB+0F	wie IOB+01.

In den Bytes IOB+08 und IOB+09 muß die Adresse eines 256 Bytes großen Puffers angegeben werden. Dies ist ein vom Benutzer zu bestimmender Speicherbereich, der die Daten des bearbeiteten Diskettensektors aufnimmt. Es ist möglich, einen der reservierten DOS-Puffer oberhalb von HIMEM zu benutzen; in dem angegebenen Beispielprogramm wird jedoch der Bereich von \$1000 bis \$10FF als Puffer verwendet.

Lesen und Schreiben eines Sektors

Das kurze Assemblerprogramm „USERWTS“ (Bild 7) dient zum Lesen oder Schreiben eines Diskettensektors unter Verwendung von RWTS. Zunächst wird mit dem Befehl JSR \$3E3 eine DOS-Routine benutzt, welche die Anfangsadresse des IOB in den Registern Y und

A zurückmeldet. Diese wird in der „Zero-Page“ zwischengespeichert und dient als Basisadresse für die Belegung der IOB-Parameter.

Das Programm erwartet Eingabewerte in den folgenden Speicherstellen:

\$19:	Nr. des Laufwerkes
\$1A:	Spur
\$1B:	Sektor
\$1C:	Kommandocode (\$01=Lesen, \$02=Schreiben).

Nach Beendigung des Programms „USERWTS“ enthält Adresse \$1D einen Return-Code, der angibt, ob bei dem vorausgegangenen Lese- oder Schreibvorgang ein Fehler aufgetreten ist (\$00 bedeutet: kein Fehler).

Ein Beispiel für die Verwendung des Programms vom Monitor aus (das Programm beginne bei Adresse \$300): Einlesen des ersten Sektors des Inhaltsverzeichnisses (Spur \$11, Sektor \$0F) vom Laufwerk Nr. 1 in den Puffer. Zunächst wird mit CALL-151 der Monitor aufgerufen, und mit

19: 01 11 0F 01

werden die Eingabewerte eingetragen.

```

1 *****
2 *
3 *      "USERWTS"      *
4 *      *              *
5 *  EIN BEISPIELPROGRAMM *
6 *  ZUR BENUTZUNG DER DOS- *
7 *  ROUTINE RWTS ZUM LESEN *
8 *  UND SCHREIBEN VON EIN- *
9 *  ZELNEN DISKETTEN-    *
10 *      SEKTOREN        *
11 *                      *
12 *****
13 ;
14      ORG $300
15 IOB   EQU $06          ; POINTER FUER IOB-ADRESSE
16 ;
17 DRIVE EQU $19          ; LAUFWERK
18 TRACK EQU $1A          ; SPUR
19 SECTOR EQU $1B         ; SEKTOR
20 COMMAND EQU $1C        ; RWTS-KOMMANDOCODE
21 RTNCODE EQU $1D        ; RETURN-CODE
22 PREG   EQU $48
23 ;
24 BUFFER EQU $1000       ; 256-BYTE-PUFFER
25 ;
26 ;
27 USERWTS JSR $3E3        ; SUCHE IOB
28         STY IOB
29         STA IOB+1       ; IOB-ADRESSE SICHERN
30 ;
31 ;  IOB-PARAMETERLISTE BELEGEN:
32 ;
33         LDY #01
34         LDA #$60
35         STA (IOB),Y     ; SLOT * 16
36         INY
37         LDA DRIVE
38
39         STA (IOB),Y
40         INY
41         LDA #00
42         STA (IOB),Y     ; VOLUME
43         INY
44         LDA TRACK
45         STA (IOB),Y
46         INY
47         LDA SECTOR
48         STA (IOB),Y
49         LDY #08
50         LDA #>BUFFER    ; NIEDERWERTIGES BYTE
51         STA (IOB),Y
52         INY
53         LDA #<BUFFER    ; HOEHERWERTIGES BYTE
54         LDY #0C
55         LDA COMMAND
56         STA (IOB),Y
57         LDY #0E
58         LDA #00
59         STA (IOB),Y
60         INY
61         LDA #$60
62         STA (IOB),Y
63 ;
64 ;  IOB-ADRESSE LADEN:
65 ;
66         LDY IOB
67         LDA IOB+1
68         JSR $3D9         ; RWTS AUFRUFEN
69         LDA #00
70         STA PREG
71         LDY #0D
72         LDA (IOB),Y
73         STA RTNCODE     ; RETURNCODE SICHERN
74         RTS

```

Bild 7. Mit diesem Programm kann ein Sektor gelesen oder beschrieben werden

Vorausgesetzt, daß das Maschinenprogramm geladen ist, kann dann der gewünschte Sektor durch

300G

eingelassen werden. Mit dem Befehl 1000.10FF wird der Inhalt des Sektors auf dem Bildschirm dargestellt. Ein eventueller Lesefehler kann durch Abfrage der Adresse \$1D erkannt werden.

Zählen freier Diskettensektoren

Es ist häufig notwendig, die Zahl der auf einer Diskette noch nicht belegten Sektoren zu bestimmen. Zu diesem Zweck verwendet man die „Volume Table of Contents“ (VTOC) auf Spur \$11, Sektor 00, die mit Hilfe des Programms „USERWTS“ in den RAM-Speicher eingelesen werden kann. Die Bytes \$38 bis \$C3 der VTOC enthalten die gewünschte Information: Jedes gesetzte Bit in diesem Bereich entspricht einem freien Sektor. Das Assemblerprogramm in Bild 8 verwendet „USERWTS“ als Unterpro-

gramm. Es benötigt als Eingabe die Nummer des Laufwerks in Speicherzelle \$19 und liefert als Ausgabe in den Adressen \$1E und \$1F die Anzahl der freien Sektoren (höherwertiges Byte in \$1F).

Vergrößerung des Speicherplatzes

Wie bereits bei der Beschreibung der Diskettenorganisation erwähnt, sind die drei Spuren 0 bis 2 für das DOS-Image reserviert, obwohl DOS die Spur 2 nur zu einem Teil tatsächlich ausnutzt. Die ungenutzten Sektoren 5 bis 15 können mit dem Programm von Bild 9 für den Benutzer verfügbar gemacht werden, ohne das auf der Diskette gespeicherte DOS-Image dabei zu beeinträchtigen.

Man gewinnt somit 11 zusätzliche Sektoren (2,75 KByte pro Diskette) für die Abspeicherung von Dateien.

Das Programm verwendet ebenfalls „USERWTS“ als Unterprogramm zum

Lesen und Schreiben der VTOC auf Spur \$11. Es ist zu beachten, daß das Programm auf jede Diskette nur einmal angewendet werden darf, um einen Verlust gespeicherter Daten zu vermeiden. Bei Auftreten eines Lese- oder Schreibfehlers wird das Programm mit zwei akustischen Signalen abgebrochen (als Eingabe wird wiederum die Nummer des Laufwerks in Zelle \$19 erwartet).

Literatur

- [1] Budge, J. H.: Inside Initialization. Apple Orchard Vol. 1, No. 2, S. 49 (1980).
- [2] Crosby, M. L.: Singin' the Disk I/O Blues. Apple Orchard Vol. 2, No. 4, S. 63 (1981).
- [3] Joepgen, H.-G.: Mehr Komfort bei Apple-Disks. mc 1981, Heft 1, S. 64.
- [4] Joepgen, H.-G.: DOS-Umschaltung beim Apple. mc 1982, Heft 6, S. 69...71.
- [5] Morris, G.: Apple II Soft Sectoring. Apple Orchard Vol. 2, No. 3, S. 26 (1981).
- [6] Worth, D., Lechner, P.: Beneath Apple DOS. Quality Software, Reseda 1982.

```

1 *****
2 *
3 *      ' F R E E '
4 *
5 * ZAEHLEN FREIER DISKETTEN-
6 *      SEKTOREN
7 *
8 *****
9 ;
10 ;
11      ORG $300
12 FREE  EQU $1E
13 ;
14 ;
15      LDA #$11
16      STA TRACK
17      LDA #00
18      STA SECTOR
19      LDA #01
20      STA COMMAND
21      JSR USERWTS
22      CLD
23      LDA #00
24      STA FREE
25      STA FREE+1      ; SUMME AUF NULL SETZEN
26      LDX #$38        ; OFFSET
27 GO     LDY #$08      ; BIT-ZAEHLER
28 BIT   CLC
29      ROL BUFFER,X    ; BIT PRUEFEN
30      BCC NEXT        ; BIT IST NICHT GESETZT
31      CLC
32      LDA FREE
33      ADC #$01
34      STA FREE
35      LDA FREE+1
36      ADC #$00        ; UEBERTRAG
37      STA FREE+1
38 NEXT  DEY
39      BNE BIT         ; 8 BITS VERARBEITET ?
40      INX             ; NAECHSTES BYTE
41      CPX #$C4        ; ENDE ?
42      BNE GO
43      RTS

```

Bild 8. Dieses Programm bestimmt die Anzahl der noch freien Sektoren einer Diskette; es verwendet die in Bild 7 abgedruckte Routine

```

1 *****
2 *
3 *      ' MOREMEM '
4 *
5 * VERGROESSERUNG DES DISK-
6 *      SPEICHERPLATZES
7 *
8 *****
9 ;
10 ;
11      ORG $300
12 BELL   EQU $FF3A      ; MONITOR-ROUTINE
13 ;
14 ;
15      LDA #$11
16      STA TRACK
17      LDA #$00
18      STA SECTOR
19      LDA #$01
20      STA COMMAND
21      JSR USERWTS
22      BCS ERROR        ; DOS-FEHLER ?
23      LDX #$40         ; BYTE-OFFSET
24      LDA #$FF
25      STA BUFFER,X     ; ALLE BITS AUF 1 !
26      INX
27      LDA BUFFER,X
28      ORA #$E0         ; BITS 5-7 AUF 1 !
29      STA BUFFER,X
30      LDA #$02         ; SCHREIBEN !
31      JSR USERWTS
32      BCS ERROR
33      RTS
34 ;
35 ERROR  LDA RTNCODE    ; RETURN-CODE PRUEFEN
36      BEQ END
37      JSR BELL         ; IM FEHLERFALL:
38      JSR BELL         ; AKUSTISCHES SIGNAL
39      END             RTS

```

Bild 9. Die Sektoren 5 bis 15 von Spur 2 sind unter DOS 3.3 normalerweise ungenutzt; mit diesem Programm kann man den Bereich ausnutzen (es wird wieder auf die Routine von Bild 7 zurückgegriffen)

Herwig Feichtinger

Apple macht Textdatei aus Speicherbereich

Nicht nur in Zusammenhang mit einem Kommunikationsprogramm kann es nützlich sein, aus einem im Speicher stehenden Text eine ASCII-Textdatei zu machen. Das folgende Programm zeigt außerdem einen Kniff, Maschinenroutinen in Basic einzubinden.

Beim Apple-II gibt es zwei Möglichkeiten, Texte auf Diskette abzuspeichern – entweder als Textdatei oder, z. B. beim Kommunikationsprogramm in Heft 1, wie ein Maschinenprogramm als Speicherauszug. Letzteres hat den Vorteil, relativ schnell zu gehen, und den Nachteil, daß ein Weiterverarbeiten einer solchen Datei von einem Basic-Programm aus problematisch ist.

Maschinenprogramm in REM-Zeile

Die hier vorgestellte Lösung zur Umwandlung eines im Speicher ab Adresse hex 1000 stehenden Textes in eine DOS-Textdatei besteht aus einem Basic-Rahmenprogramm und einer kurzen Maschinenroutine, die innerhalb einer REM-Basic-Zeile steht und somit nicht getrennt von Disk geladen werden muß. Die Eingabe erfolgt so: Zunächst tippt man das Applesoft-Programm aus Bild 1 ein, wobei in Zeile 1 hinter REM nur etwa 50 Doppelpunkte (oder beliebige andere Zeichen) stehen. Dieser Platz ist dem Maschinenprogramm in Bild 2 vorbehalten. Man gibt es ein, indem man mit CALL-151 zum Monitor geht und dann ab Adresse 0806 die einzelnen Bytes eingibt (0806: A9 FF 8D...). Nach Return, CTRL-C und nochmals Return ist man wieder in Basic. Wenn man nun in Basic LIST eingibt, sollte man sich über den merkwürdigen Inhalt der REM-Zeile nicht wundern. Denn Basic interpretiert die Maschinenprogramm-Bytes nun zum Teil als reservierte Basic-Worte, wie das auch in Bild 1 zu sehen ist. Direkt diese Worte in Basic statt das Maschinenprogramm vom Monitor aus einzugeben funk-

tiert übrigens nicht, weil nicht alle Bytes überhaupt auf dem Bildschirm erscheinen! Mit SAVE MEMTXT kann man nun Maschinen- und Basic-Programm auf einmal auf Diskette abspeichern. Zeile 15 sollte übrigens entfallen, wenn Bit 7 beim Text im Speicher nicht gesetzt ist.

Vor dem Start des Programms muß der Text bereits ab Adresse hex 1000 im Speicher stehen (man kann ihn, falls er auf Disk steht, dorthin mit BLOAD NAME,A\$1000 bringen). Nach RUN wird man nach dem gewünschten Namen der zu erzeugenden Textdatei gefragt. Der Rest funktioniert automatisch, und zwar dank des Maschinenprogramms vergleichsweise schnell.

Das Komma-Problem

Die Maschinenroutine ersetzt in der abgedruckten Version alle Kommas und Doppelpunkte durch Strichpunkte, um zu vermeiden, daß man beim späteren Einlesen der Textdatei mit dem Basic-Befehl INPUT die Fehlermeldung EXTRA IGNORED erhält. Wenn man aber z. B. mit dem Kommunikationsprogramm aus Heft 1 ein Basic-Programm empfangen hat (Download), so darf diese Umcodierung nicht stattfinden. Im Maschinenprogramm muß man dann die Bytes an den Adressen 0821 und 0825 durch hex BB ersetzen. Eine damit erzeugte Textdatei, die ein Basic-Programm enthält, kann man anschließend leicht mit EXEC NAME in ein „richtiges“ Basic-Programm umwandeln (vorher NEW nicht vergessen!).

```
1REMSPEED=hPLOTSPEED=PLOTOUT OF DATA=
OUT OF DATAIFh= END-RUNOVERFL
OW-PRINT=SPEED=CONT RETURN WITHOUT GOSUBhL
:~::~:
10HOME:PRINT"Umwandlung des Speicherinhalts ab $1000
11PRINT"in eine ASCII-Textdatei":VTAB5
12HIMEM:4095:D$=CHR$(4)
15IFPEEK(4096)<128THENPRINT"Kein Text!":END
20INPUT"Dateiname: ";N$
30PRINTD$;"OPEN";N$:PRINTD$;"DELETE";N$
40PRINTD$;"OPEN";N$:PRINTD$;"WRITE";N$
50CALL2054:REM $806
60PRINT:PRINT"END":PRINTD$;"CLOSE";N$
70PRINT"Fertig!":END
```

Bild 1. Basic-Programm. Statt der merkwürdigen Dinge hinter REM in Zeile 1 sind allerdings zunächst etwa 50 Doppelpunkte einzugeben – hier steht später das Maschinenprogramm

0806-	A9 FF	LDA	£\$FF
0808-	8D 19 08	STA	\$0819
080B-	A9 0F	LDA	£\$0F
080D-	8D 1A 08	STA	\$081A
0810-	EE 19 08	INC	\$0819
0813-	DO 03	BNE	\$0818
0815-	EE 1A 08	INC	\$081A
0818-	AD FF FF	LDA	£FFF
081B-	DO 01	BNE	\$081E
081D-	60	RTS	
081E-	09 80	ORA	£\$80
0820-	C9 AC	CMP	£\$AC
0822-	FO 04	BEQ	\$0828
0824-	C9 BA	CMP	£\$BA
0826-	DO 02	BNE	\$082A
0828-	A9 BB	LDA	£\$BB
082A-	20 ED FD	JSR	\$FDED
082D-	4C 10 08	JMP	\$0810

Bild 2. Maschinenprogramm als Disassembler-Listing

Wolfgang Ebner

Apple-Grafik füllt eine DIN-A4-Seite

Verschiedene Druckerschnittstellen für den Apple-II (bzw. den weitgehend kompatiblen Basis-108) enthalten bereits Routinen zur Ausgabe einer Grafikseite. Auf dem Papier erscheinen die Zeichnungen meist ziemlich klein. Mit dem beschriebenen Maschinenprogramm können Sie Ihre Punktgrafiken so zu Papier bringen, daß ein Bildschirminhalt genau einer DIN-A4-Seite entspricht. An Hardware benötigt der Computer den Epson-Drucker MX-82 und eine Parallelschnittstelle.

Nachdem sich gezeigt hatte, daß ein Basic-Programm mit der erwarteten geringen Geschwindigkeit arbeitet, wurde ein Assemblerprogramm (Bild 1) geschrieben, das in seinen prinzipiellen Möglichkeiten dem Grafik-Interface von Epson entspricht. Daran schloß sich ein weiterer Schritt an: Der MX-82 kann bei einfacher Druckdichte horizontal 576 Punkte

drucken, das entspricht exakt dem dreifachen Wert der Apple-Grafik in vertikaler Richtung. Durch Drehen des Bildes um 90 Grad läßt sich somit jeder Punkt der Apple-Grafik als 3*3-Matrix auf dem Drucker darstellen: Dies entspricht einer Vergrößerung auf die neunfache Fläche. Darüber hinaus bietet das Programm die Optionen „Invertieren“, „Druck der

Bildschirmseite zwei“ und „doppelte Druckdichte“ (ohne das Bild auf die Hälfte zu stauchen, wie das beim Epson-Interface der Fall ist).

Um das Programm möglichst kurz zu halten, werden die Optionen nicht mit Hilfe eines Mode-Registers übergeben, sondern ein kurzes Basic-Programm ruft den Assembler-Teil auf und modifiziert das Programm entsprechend den Optionen. Danach kann der Ausdruck mit dem „&“-Befehl des Applesoft-Interpreters gestartet werden bzw. mit „CALL 768“. Das Programm belegt die dritte Speicherseite vor den DOS-Vektoren (\$300...\$3CD) und kann somit weder von Basic überschrieben werden, noch kommt es mit DOS oder Basic in Konflikt.

Um das Programm benutzen zu können, ist also folgendes erforderlich: ein Apple-II bzw. ein dazu kompatibler Computer; ein beliebiges Parallel-Interface (es werden nur die Hardware-Adressen „Strobe“ (\$C1C1) und „Output“ (\$C090) vorausgesetzt); ein Epson-MX-82 (ein Epson-MX-100 tut es auch; der MX-80 hat leider nur 480 Druckpunkte, so daß auch nur 160 der 192 Grafikpunkte ausgedruckt werden können; durch Modifizieren des Programms kann z. B. der Bereich von Grafikzeile 16 bis 176 ausgedruckt werden, was oftmals ausreicht); der Applesoft-Interpreter (für die

1	*****	0333: 20 B8 03 42	JSR	DECTL
2	*** HGR DUMP *****	0336: A5 01 43	LDA	COUNT
3	*** FUER EPSON MX82 *****	0338: C9 02 44	NOCA	CMP #2
4	*** W.EBNER 2/83 *****	033A: D0 E8 45	BNE	LOOP2
5	*****	033C: A2 03 46	LDX	#3
6	HPAG EQU \$E6 ;HGR-PAGE (\$20/\$40)	033E: A5 00 47	LDA	BYTE
7	HBASL EQU \$26 ;HGR-SPEICHERADRESSE	0340: EA 48	NOP	
8	HBASH EQU HBASL+1 ; ^ HIGH BYTE	0341: EA 49	NOP	
9	HMASK EQU \$30 ;FARBBITMASKE FUER HGR	0342: 20 B6 03 50	OUT	JSR POUT1
10	XL EQU \$E0 ;X-KOORDINATE LOW	0345: CA 51	DEX	
11	XH EQU XL+1 ;X-KOORDINATE HIGH	0346: D0 FA 52	BNE	OUT
12	YL EQU \$E2 ;Y-KOORDINATE	0348: E6 E2 53	INC	YL
13	BYTE EQU \$0 ;DRUCKBYTE	034A: A5 E2 54	LDA	YL
14	COUNT EQU \$1 ;ZAEHLER	034C: C9 C0 55	CMP	#192
15	*****	034E: D0 C2 56	BNE	LOOP1
16	ORG \$300	0350: C6 E0 57	R	DEC XL
17	*****	0352: 20 B8 03 58	JSR	DECTL
0300: 20 6F 03 18	START JSR INIT ;GRAPHIC/DRUCKER INITIALISIEREN	0355: A9 FF 59	NOC	LDA #\$FF
0303: A9 01 19	LDA #1	0357: C5 E1 60	CMP	XH
0305: 85 E1 20	STA XH	0359: D0 B0 61	BNE	LOOP
0307: A9 15 21	LDA #21	035B: A9 FD 62	LDA	#\$FD
0309: 85 E0 22	STA XL ;X-KOORDINATE INITIALISIEREN	035D: C5 E0 63	CMP	XL
030B: A9 00 23	LOOP LDA #0	035F: D0 AA 64	BNE	LOOP
030D: 85 E2 24	STA YL	0361: 20 C3 03 65	ENDE	JSR CRLF
030F: 20 BF 03 25	JSR PINIT ;DRUCKERGRAPHIK EINSCHALTEN	0364: A9 18 66	LDA	#27
0312: A9 00 26	LOOP1 LDA #0	0366: 20 B6 03 67	JSR	POUT1
0314: 85 01 27	STA COUNT ;ZAEHLER = 0	0369: A9 32 68	LDA	#\$32
0316: 85 00 28	STA BYTE ;DRUCKBYTE = 0	036B: 20 B6 03 69	JSR	POUT1
0318: E6 E0 29	INC XL	036E: 60 70	RTS	
031A: E6 E0 30	INC XL ;X-KOORDINATE KORRIGIEREN	71	*****	
031C: A9 01 31	LDA #1	72	*****	
031E: C5 E0 32	CMP XL ;XL=1 ?	036F: A9 20 73	INIT	LDA #20
0320: D0 02 33	BNE LOOP2 ;NEIN	0371: 85 E6 74	STA	HPAG
0322: E6 E1 34	INC XH ;UEBERLAUF	0373: A9 1B 75	LDA	#27
0324: 06 00 35	LOOP2 ASL BYTE ;DRUCKBYTE NACH LINKS SCHIEBEN	0375: 20 B6 03 76	JSR	POUT1
0326: 06 00 36	ASL BYTE	0378: A9 41 77	LDA	#65
0328: 06 00 37	ASL BYTE	037A: 20 B6 03 78	JSR	POUT1
032A: 20 A4 03 38	JSR HSCRN ;PUNKT ABFRAGEN	037D: A9 06 79	LDA	#6
032D: 05 00 39	ORA BYTE	037F: 20 B6 03 80	JSR	POUT1
032F: 85 00 40	STA BYTE ;DRUCKBYTE AKTUALISIEREN	0382: 20 C3 03 81	JSR	CRLF
0331: E6 01 41	INC COUNT			

Bild 1. Assemblerlisting der Grafikroutine. Wie man ein solches Programm in den Computer bringt und abspeichert, ist in [1] beschrieben

Neu! DISTAR-Laufwerk für Apple II oder ähnliche

- * Apple II-kompatibel
- * halbspurfähig
- * 40 Spuren, 163 KB
- * Spur-0-Erkennung
- * Direktantrieb
- * Stahlband-Positionierung
- * Kabel für Disk-II-Controller
- * kann DOS 3.3, PASCAL, CP/M



Händler-Anfragen erwünscht!

498. inkl.
MwSt.

Distributor:
Dipl.-Ing. R. Springmann

Stöckener Straße 199,
3000 Hannover 21

Telefon 05 11/79 11 11,
Telex 921 466 comps d

Bildpunktabfrage werden Routinen des Interpreters benutzt).

Der benutzte Assembler interpretiert Zahlen ohne Präfix als Dezimalzahlen, mit „\$“ als hexadezimal, mit „%“ binär.

Das Hauptprogramm (Zeile 18...70) besteht aus einer dreifach geschachtelten Schleife, in der jeder Bildpunkt abgefragt und das Druckbyte aufgebaut wird.

Die Unterprogramme wurden teilweise nur der Übersichtlichkeit halber ausgelagert (INIT, PINIT, und HSCRN werden nur einmal vom Hauptprogramm aus

aufgerufen). Die Kommentare zeigen die Stellen, an denen das Basic-Programm die Optionen einschreibt (zu diesem Zweck dienen auch die NOPs in Zeile 48 und 49). Abgespeichert wird das Assemblerprogramm mit „BSAVE HGR DUMP.

BIN,\$A300,\$L\$CE“. Es läßt sich nun mit „BRUN“ starten, bzw. mit „CALL 768“, wenn es schon geladen ist.

Um die angesprochenen Optionen einfach zugänglich zu machen, ist es jedoch besser, das in Bild 2 abgedruckte Basic-Programm zum Starten zu benutzen. Es wird eingetippt und mit „SAVE HGR DUMP“ gespeichert. Nach dem Pro-

grammstart lädt es zuerst den Maschinenspracheteil, dann zeigt es ein kleines Menü an: Zum Auswählen einer Option sind die entsprechende Zahl und Return einzugeben. Die Programmzeilen 350...370 nehmen dann den entsprechenden „Patch“ vor. Zum Verlassen des Programms ist eine „9“ einzugeben.

```

03B5: 60      82      RTS      ;ENDE INIT
          83      *****
03B6: 2C C1 C1 84 POUT1 BIT  %C1C1 ;TEST PRINTER STROBE
03B9: 30 FB 85      BMI POUT1 ;NOT READY
03B8: 8D 90 C0 86 STA  %C090 ;OUTPUT TO PRINTER
03BE: 60      87      RTS      ;ENDE POUT1
          88      *****
03BF: A9 1B 89      PINIT LDA  #27
0391: 20 B6 03 90 JSR  POUT1 ;SENDET ESC
0394: A9 48 91      LDA  #$4B ;DOUBLE DENSITY PATCH HIER
0396: 20 B6 03 92 JSR  POUT1 ;SENDET "K"
0399: A9 40 93      LDA  #64 ;DOUBLE DENSITY PATCH HIER
039B: 20 B6 03 94 JSR  POUT1
039E: A9 02 95      LDA  #2 ;DOUBLE DENSITY PATCH HIER
03A0: A9 03 96      JSR  POUT1 ;SENDET "576 CHAR."
03A3: 60      97      RTS      ;ENDE PINIT
          98      *****
03A4: A6 E0 99      HSCRN LDX  XL
03A6: A4 E1 100     LDY  YL
03A8: A5 E2 101     LDA  YL ;X UND Y KOORDINATE LADEN
03AA: 20 11 F4 102 JSR  %F411 ;POSITIONIERT HIRESCURSOR
03AD: A5 30 103     LDA  HMASK ;PUNKTMASKE LADEN
03AF: 29 7F 104     AND  #$7F ;MSB LOESCHEN
03B1: 31 26 105     AND  (HBASL),Y ;TESTET PUNKT
03B3: F0 02 106     BEQ  HSCRN1 ;PUNKT AUS
03B5: A9 07 107     LDA  #%111 ;PUNKT EIN
03B7: 60      108     HSCRN1 RTS      ;ENDE HSCRN
          109     *****
03B8: C6 E0 110     DECBXL DEC  XL
03BA: A9 FF 111     LDA  #$FF
03BC: C5 E0 112     CMP  XL ;UEBERLAUF ?
03BE: D0 02 113     BNE  RTS ;NEIN
03C0: C6 E1 114     DEC  XH ;XH KORRIGIEREN
03C2: 60      115     RTS      RTS
          116     *****
03C3: A9 0D 117     CRLF LDA  #D
03C5: 20 B6 03 118 JSR  POUT1 ;CR AUSGEBEN
03C8: A9 0A 119     LDA  #A
03CA: 20 B6 03 120 JSR  POUT1 ;LF AUSGEBEN
03CD: 60      121     RTS      ;ENDE CRLF
          122     *****

```

```

100 REM HGR DUMP für Epson MX82; W.Ebner 2/83
105 REM *****
110 PRINT CHR$(4) "BLOAD HGR DUMP.BIN"
120 TEXT : HOME
130 PRINT TAB(10) "H G R - D U M P"
140 PRINT TAB(10) "*****"
150 PRINT : PRINT
160 PRINT "Optionen : "
170 PRINT : PRINT
180 PRINT "1 - Doppelte Druckdichte"
190 PRINT
200 PRINT "2 - Invertieren"
210 PRINT
220 PRINT "3 - Bildschirmseite 2"
230 PRINT : PRINT
240 PRINT "9 - Programmende"
250 VTAB 23
260 PRINT SPC(30);
270 HTAB 1
280 INPUT "Eingabe: ";Z$
290 Z = VAL (Z$)
300 IF Z < 4 AND Z > 0 THEN 350
310 IF Z > 9 THEN 250
320 HOME : PRINT "HGR DUMP geladen."
    Programmstart mit '&'.
330 POKE 1014,0: POKE 1015,3
340 PRINT : END
350 IF Z = 1 THEN POKE 829,6:
    POKE 917,76: POKE 922,128: POKE 927,4
360 IF Z = 2 THEN POKE 832,73: POKE 833,63
370 IF Z = 3 THEN POKE 880,64
380 GOTO 250

```

Bild 2. Dieses Basic-Programm lädt die Grafikroutine von der Diskette (HGR DUMP, BIN) und nimmt die Auswahl der Optionen entgegen

In Zeile 330 wird der &-Vektor eingerichtet, und von da an kann der Ausdruck mit „&“ im Direktmodus oder von einem Programm aus gestartet werden.

Literatur

[1] Hofer, Rudolf: Vom Umgang mit Apple-Maschinenprogrammen. mc 1983, Heft 3, Seite 81.

Dipl.-Phys. H. M. Ihme

Apple-Ted: Ein komfortabler Texteditor

Wenn Sie mit Ihrem Apple-II Briefe oder andere Texte bearbeiten wollen, dann ist Apple-Ted das richtige Programm für Sie. Es stellt einen zeigerorientierten Editor dar, der sehr komfortable Möglichkeiten bietet. Apple-Ted läuft auf 48-KByte-Maschinen unter dem Betriebssystem DOS 3.3.

Als der Autor dieses Beitrags im Funkschau-Sonderheft Nr. 31 [2] einen 6502-Texteditor für den KIM beschrieb, behauptete er damals kühn, eine Anpassung an andere Systeme sei leicht möglich, da nur zwei Befehle geändert wer-

den müßten. Nach einer etwas eingehenderen Beschäftigung mit verschiedenen Computern kann man diese Behauptung nicht mehr aufrecht erhalten. Es zeigt sich nämlich, daß fast jede Firma ihr eigenes Tastensüppchen kocht, dessen

Kochrezepte sie manchmal so geheim zu halten sucht, daß ein Nachwürzen unmöglich wird. Deshalb wird hier der damals veröffentlichte Texteditor als Apple-Ted noch einmal vorgestellt – voll angepaßt an den Apple-II, von einigen Restfehlern befreit und um einige (manchmal vermißte) Befehle erweitert (Bild 1). Da alles damals Geschriebene auch heute noch Gültigkeit hat, sollen nur die Veränderungen und Erweiterungen besprochen werden; eine Bedienungsanleitung findet man im Kasten. Das Maschinenprogramm belegt im Apple den Bereich \$800...\$12FF. Die Adressen für die verschiedenen Buffer stehen in den Speicherzellen von \$806...\$811 und können leicht geändert werden. Im hier veröffentlichten Programm sind die Speicherbereiche:

Eingangsspeicher \$9000...\$95FF
Hauptspeicher \$2000...\$8FFF
Hilfsspeicher \$1300...\$1AFF

Der laufende Zeiger wird als blinkendes Dach dargestellt, alle eingegebenen Steuerzeichen als inverse Buchstaben.

```

0800- 4C 2F 0B 4C 54 0B 90 00
0808- 95 FF 20 00 8F FF 13 00
0810- 1A FF 46 C3 A0 B1 B9 B8
0818- B2 A0 C2 D9 A0 C8 AE CD
0820- AE C9 CB CD C5 00 A5 99
0828- 85 80 A5 9A 85 B1 60 E6
0830- 84 D0 02 E6 85 60 A5 86
0838- 85 8E A5 87 85 8F 60 A5
0840- 8E 85 86 A5 8F 85 87 60
0848- 20 A7 09 20 FF 09 B0 0E
0850- C5 B0 10 0B 20 E5 09 B0
0858- 05 20 77 09 90 F6 60 20
0860- F1 09 B0 FA 20 65 09 90
0868- F6 60 A0 0B B9 A1 12 20
0870- ED FD 88 10 F7 20 3A FF
0878- 4C AA 0B 20 C8 09 A5 84
0880- 85 99 A5 85 85 9A A0 FF
0888- 84 98 38 A5 82 E5 80 85
0890- 88 A5 83 E5 81 85 89 18
0898- A5 88 65 84 85 86 48 A5
08A0- 89 65 85 85 87 48 38 AD
08A8- 0D 08 E5 86 AD 0C 08 E5
08B0- 87 90 B7 E6 98 E6 88 E6
08B8- 89 38 A5 84 E5 80 A5 85
08C0- E5 81 A2 00 90 02 A2 02
08C8- A1 80 81 84 90 14 C6 82
08D0- 98 45 82 D0 02 C6 83 C6
08D8- 86 98 45 86 D0 02 C6 87
08E0- 80 06 20 32 09 20 2F 08
08E8- C6 88 D0 02 C6 89 D0 D2
08F0- 20 D1 09 A5 98 10 05 68
08F8- 68 38 B0 07 68 85 83 68
0900- 85 82 18 A2 00 60 20 4F
0908- 09 B0 0E A1 86 48 20 39
0910- 09 68 C9 A3 D0 02 18 60
0918- 38 60 A2 03 A9 00 95 92
0920- CA 10 FB E8 95 98 60 38
0928- A5 80 E5 82 A5 81 E5 83
0930- B0 06 E6 80 D0 02 E6 81
0938- 60 38 A5 86 ED 09 08 E9
0940- 02 A5 87 ED 08 08 B0 06
0948- E6 86 D0 02 E6 87 60 38
0950- AD 07 08 E5 86 AD 06 08
0958- E5 87 B0 08 A5 86 D0 02
0960- C6 87 C6 86 60 38 A5 88
0968- E5 82 A5 89 E5 83 B0 06
0970- E6 88 D0 02 E6 89 60 38
0978- AD 08 08 E5 88 AD 0A 08
0980- E5 89 B0 08 A5 88 D0 02
0988- C6 89 C6 88 60 20 65 09
0990- B0 07 A1 88 C9 8D D0 F5
0998- 18 60 20 77 09 B0 07 A1
09A0- 88 C9 8D D0 F5 18 60 A5
09A8- 80 85 88 A5 81 85 89 60
09B0- A5 88 85 80 A5 89 85 81
09B8- 60 A2 01 E5 88 84 80 95
09C0- 80 94 88 CA 10 F5 E8 60
09C8- A5 80 85 8A A5 81 85 88
09D0- 60 A5 8A 85 80 A5 88 85
09D8- 81 60 A5 80 45 88 D0 04
09E0- A5 81 45 89 60 20 FF 09
09E8- B0 06 E6 91 D0 02 E6 90
09F0- 60 20 FF 09 B0 FA A5 91
09F8- D0 02 C6 90 C6 91 60 A5
0A00- 90 05 91 F0 02 18 60 38
0A08- 60 86 FA 20 A7 09 20 FF
0A10- 09 90 07 20 9A 09 B0 ED
0A18- 90 ED 24 90 30 0C 20 F1
0A20- 09 B0 06 20 8D 09 4C 1E
0A28- 0A 60 20 9A 09 B0 D6 20
0A30- E5 09 90 F6 60 A1 86 38
0A38- E9 B0 90 05 C9 0A 90 02
0A40- 18 60 48 B5 93 95 92 E8
0A48- E0 04 90 F7 68 85 95 A2
0A50- 00 20 39 09 38 60 20 5F
0A58- 0A 20 8E FD 4C C1 0E B9
0A60- 01 00 85 90 B9 00 00 B5
0A68- 91 05 90 D0 03 4C DA FD
0A70- FB A0 10 8A A2 02 95 92
0A78- CA 10 FB 06 91 26 90 90
0A80- 0B A2 02 B5 92 69 00 95
0A88- 92 CA 10 F7 88 F0 0F 18
0A90- A2 02 B5 92 75 92 95 92
0A98- CA 10 F7 4C 7B 0A D8 84
0AA0- 9B B9 92 00 D0 04 46 98
0AA8- 90 05 20 DA FD E6 98 C8
0AB0- C0 03 90 ED 60 18 A5 91
0AB8- 75 92 85 91 A5 90 69 00
0AC0- 85 90 60 A0 03 8A 85 90
0AC8- 85 91 85 96 20 B5 0A A5
0AD0- 91 85 97 A5 90 85 96 06
0AD8- 91 26 90 06 91 26 90 18
0AE0- A5 91 65 97 85 91 A5 90
0AE8- 65 96 85 90 06 91 26 90
0AF0- E8 88 D0 D8 20 B5 0A A2
0AF8- 00 A5 98 D0 01 60 38 8A
0B00- E5 91 85 91 8A E5 90 85
0B08- 90 60 AD 08 08 85 82 AD
0B10- 0A 08 85 83 A2 00 60 86
0B18- FA AD 08 08 85 80 AD 0A
0B20- 08 85 81 60 AD 07 08 85
0B28- 86 AD 06 08 85 87 60 20
0B30- 58 FC A2 19 AD 25 08 D0
0B38- 02 A2 0A 20 4A F9 A0 0E
0B40- B9 AD 12 20 33 11 88 10
0B48- F7 20 8E FD 20 A4 10 20
0B50- EE 10 D0 12 20 0A 08 A1
0B58- 82 C9 FF F0 09 E6 82 D0
0B60- 02 E6 83 4C 57 0B D8 BA
0B68- 86 FD A9 40 8D 5E AA A2
0B70- 21 B5 80 9D 04 03 CA 10
0B78- F8 E8 A9 07 A0 C3 2C 25
0B80- 08 30 04 A9 F0 A0 FD BD
0B88- 00 03 8C 01 03 AD F2 03
0B90- 8D 02 03 AD F3 03 BD 03
0B98- 03 A9 AA 8D F2 03 A9 08
0BA0- 8D F3 03 20 6F FB A9 08

```

Bild 1. Hexlisting des Editors. Ein kommentiertes Assemblerlisting und eine Diskette mit Objekt- und Quellcode können vom Franzis-Software-Service bezogen werden. Der Quellcode liegt im Format des Assemblers „Lisa“ vor

Das Apple-Sonderheft

```

0B4B- 85 F8 A6 FD 9A AD 00 03
0BB0- 85 36 AD 01 03 85 37 20
0BB8- EA 03 A2 00 20 17 08 A9
0BC0- 9B 85 F9 86 FC 20 39 11
0BC8- A9 BF 20 33 11 20 24 08
0BD0- 20 2A 11 C9 C0 D0 0B 20
0BD8- 39 09 20 2A 11 85 F9 4C
0BE0- D0 0B A4 F9 C0 9B D0 09
0BE8- C9 98 D0 05 86 FA 4C BF
0BF0- 0B C0 9B D0 36 C9 92 D0
0BF8- 32 86 FA A5 FE 48 20 36
0C00- 08 20 4F 09 B0 08 A1 86
0C08- C9 8D D0 F5 F0 03 20 8E
0C10- FD A5 86 C5 BE A5 87 E5
0C18- 8F B0 0A A1 86 20 33 11
0C20- 20 39 09 90 EC 68 85 FE
0C28- 4C D0 0B C0 9B D0 2A C9
0C30- 88 D0 26 20 4F 09 B0 87
0C38- A1 86 C9 89 D0 0F 38 A5
0C40- FE E5 FB E5 FB 85 FE 20
0C48- 82 11 4C D0 08 C6 FE C6
0C50- FE C6 FA 20 33 11 4C D0
0C58- 0B 81 86 C9 9B F0 08 20
0C60- 3F 09 B0 16 4C D0 0B 20
0C68- 49 09 90 03 4C C5 0B A1
0C70- 86 C9 9B F0 10 20 39 09
0C78- 90 E5 A9 9B 81 86 20 39
0C80- 09 A9 9B 81 86 20 8E FD
0C88- 86 8C 86 8D 20 24 0B 20
0C90- 1A 09 A1 86 C9 AD D0 05
0C98- E6 98 20 39 09 A1 86 29
0CA0- B0 FB 20 C3 0A A1 86 29
0CA8- DF C9 D8 D0 16 A5 98 F0
0CB0- 03 20 FE 0A A5 90 85 BD
0CB8- A5 91 85 8C A5 86 85 9B
0CC0- 4C AC 0E C9 CA D0 06 20
0CC8- 17 0B 4C C1 0E C9 DA D0
0CD0- 0B A5 82 85 80 A5 83 85
0CD8- 81 4C C1 0E CD 22 08 D0
0CE0- 1C A2 21 BD 04 03 95 B0
0CE8- CA 10 F8 AD 02 03 8D F2
0CF0- 03 AD 03 03 BD F3 03 20
0CF8- 6F FB 4C BF 9D C9 D4 D0
0D00- 03 4C CA 0E CD 1B 08 D0
0D08- 07 A5 91 85 FB 4C C1 0E
0D10- CD 23 0B D0 06 20 48 0B
0D18- 4C 60 D0 C9 CA D0 03 4C
0D20- 8C 10 CD 21 0B D0 03 4C
0D28- 75 0F C9 CC D0 3B 20 39
0D30- 09 A1 86 29 DF C9 CF D0
0D38- 06 20 1F 12 4C 54 0B 20
0D40- 4F 09 A1 80 C9 BD D0 15
0D48- 38 AD 0F 08 E5 80 AD 0E
0D50- 0B E5 81 B0 0B A5 80 D0
0D58- 02 C6 81 C6 80 20 B8 10
0D60- 20 B0 09 4C C1 0E C9 CB
0D68- D0 03 4C 70 10 C9 C6 D0
0D70- 06 20 F1 0F 4C BF 0C CD
0D78- 13 0B D0 03 4C 39 10 C9
0D80- D6 D0 32 20 C8 09 20 17
0D88- 0B 8A 85 9E 85 9F 8A 85
0D90- 84 85 85 A1 80 C9 BD D0
0D98- 0A 20 2F 08 20 27 09 90
0DA0- F2 B0 04 C9 BC D0 F5 20
0DA8- 6D 12 20 27 09 90 DF 20
0DB0- D1 09 4C C1 0E C9 CE D0
0DB8- 06 20 B1 11 4C C1 0E CD
0DC0- 24 0B D0 15 A9 A4 20 33
0DC8- 11 A5 83 20 DA FD A5 82
0DD0- 20 DA FD 20 8E FD 4C C1
0DD8- 0E CD 1A 0B D0 4B A5 90
0DE0- 85 9E A5 91 85 9F 20 39
0DE8- 09 A1 86 29 DF C9 C3 D0
0DF0- 06 20 05 11 4C C1 0E C9
0DF8- D4 D0 09 20 05 11 20 AC
0E00- 10 4C 75 10 C9 CB D0 06
0E08- 20 EE 10 4C C1 0E C9 D2
0E10- D0 F9 20 C0 10 A1 A0 C9
0E18- FF D0 03 4C C1 0E 20 33
0E20- 11 20 D4 10 90 EF C9 D0
0E28- D0 3D 20 39 09 A1 86 29
0E30- DF C9 D2 D0 2F 20 39 09
0E38- 20 09 09 80 27 A1 86 C9

```

```

0E40- B6 B0 21 C9 B0 90 1D F0
0E48- 09 18 29 3F 69 90 85 37
0E50- 86 36 3B 66 FC 20 4F 09
0E58- 20 4F 09 20 4F 09 20 EA
0E60- 03 4C CA 0E 4C 67 0F C9
0E68- 89 D0 0E 20 4F 09 90 06
0E70- 20 5C 09 20 5C 09 4C 75
0E78- 0F C9 C0 D0 03 4C C1 0E
0E80- C9 D3 D0 11 20 39 09 A1
0E88- 86 29 DF C9 C1 D0 D5 20
0E90- F6 11 4C 54 0B C9 84 D0
0E98- 06 20 4A 12 4C 54 0B C9
0EA0- 9B D0 1E 20 39 09 A1 86
0EA8- C9 9B D0 18 A5 8C 05 8D
0EB0- F0 15 A5 8C D0 02 C6 BD
0EB8- C6 8C 20 24 0B A5 9B 85
0EC0- 86 20 39 09 4C BF 0C 4C
0EC8- BF 08 20 C8 09 20 06 09
0ED0- B0 11 20 17 0B A1 80 C9
0ED8- FF F0 33 20 33 11 20 27
0EE0- 09 90 F2 20 FF 09 90 4B
0EE8- 86 FA 20 A7 09 20 9A 09
0EF0- B0 03 20 65 09 20 DA 09
0EF8- D0 05 A9 5E 20 ED FD A1
0F00- 8B C9 FF F0 09 4B 20 33
0F08- 11 68 C9 8D D0 E4 20 D1
0F10- 09 20 8E FD AD 00 03 85
0F18- 36 AD 01 03 85 37 20 EA
0F20- 03 24 FC 10 09 20 39 09
0F28- 20 39 09 20 39 09 86 FC
0F30- 4C C1 0E 24 90 10 1B 20
0F38- 09 0A 20 DA 09 F0 CF A1
0F40- 8B C9 FF F0 C9 20 33 11
0F48- 20 65 09 20 DA 09 D0 EF
0F50- F0 BC 20 09 0A 20 B9 09
0F58- 4C 3F 0F 20 7B 08 20 3F
0F60- 0B 4C 26 08 20 3F 0B A0
0F68- 0F B9 CB 12 20 ED FD 88
0F70- 10 F7 4C 75 08 3B A5 80
0F78- E9 01 85 84 A5 81 E9 00
0F80- 85 85 20 39 09 A1 86 C9
0F88- 82 D0 36 A5 F9 C9 9B D0
0F90- 30 20 36 08 20 C0 10 20
0F98- 2F 0B A1 A0 C9 FF F0 05
0FA0- 20 D4 10 90 F2 20 7B 08
0FA8- 20 C0 10 A1 A0 C9 FF F0
0FB0- 0A 81 80 20 D4 10 20 27
0FB8- 09 90 F0 20 3F 0B 4C EB
0FC0- 0F 20 4F 09 20 36 08 20
0FC8- 2F 08 20 39 09 B0 95 A1
0FD0- 86 C5 F9 D0 F2 20 7B 08
0FD8- 20 3F 08 20 39 09 A1 86
0FE0- C5 F9 F0 07 81 80 20 27
0FE8- 09 90 F0 20 26 08 4C 8F
0FF0- 0C 20 39 09 20 C8 09 A0
0FF8- 00 84 9B 81 86 C9 9B F0
1000- 1A C1 80 F0 0A 20 D1 09
1008- 20 27 09 B0 1E 90 E5 A9
1010- 01 85 9B C8 20 27 09 B0
1018- 12 90 E0 A5 9B F0 0C 1B
1020- 9B 65 86 85 86 8A 65 87
1028- 85 87 60 A0 0E 89 BC 12
1030- 20 ED FD 88 10 F7 4C AA
1038- 0B A9 9B 85 F9 20 F1 0F
1040- 84 9B 3B A5 80 E5 9B 85
1048- 84 A5 81 E9 00 85 85 20
1050- 36 08 20 5B 0F 20 39 09
1058- A1 86 C9 9B D0 03 4C C8
1060- 0B 20 4F 09 4C 75 0F 20
1068- A4 10 20 17 0B 4C C1 0E
1070- 20 06 09 90 F2 20 B5 10
1078- 20 CD 10 20 36 0B A5 88
1080- 85 84 A5 89 85 85 20 5B
1088- 0F 4C C1 0E 20 95 10 20
1090- 4B 0B 4C 7B 10 8A 85 99
1098- 24 90 30 07 20 FF 09 B0
10A0- 02 E6 99 60 20 0A 0B A9
10A8- FF 81 82 60 A5 9E 85 90
10B0- A5 9F 85 91 60 20 95 10
10B8- 20 09 0A 09 F7 4C 65 09
10C0- 86 FA AD 0F 08 85 A0 AD
10C8- 0E 0B 85 A1 60 46 99 90
10D0- FB 4C B9 09 3B A5 A0 ED

```

```

10D8- 11 0B A5 A1 ED 10 0B B0
10E0- 07 E6 A0 D0 02 E6 A1 60
10E8- 20 F1 10 4C 6A 0B 20 C0
10F0- 10 A9 FF 81 A0 60 20 C0
10F8- 10 A1 A0 C9 FF F0 F6 20
1100- D4 10 4C F9 10 20 C8 09
1108- 20 F6 10 20 B5 10 20 CD
1110- 10 20 DA 09 F0 0E A1 8B
1118- 81 A0 20 D4 10 20 65 09
1120- B0 02 90 ED 20 F1 10 4C
1128- D1 09 20 0C FD 4B 20 33
1130- 11 68 60 85 9C C9 8D D0
1138- 0A 20 8E FD A2 00 86 FE
1140- 86 FA 60 C9 8B F0 3B C9
1148- 9F F0 37 24 FC 30 22 C9
1150- 9B D0 04 A9 24 D0 1B 80
1158- 18 2C 25 0B 10 07 4B A9
1160- DE 20 6F 11 68 1B 69 40
1168- 2C 25 0B 30 02 29 3F 85
1170- 9C E6 FA 24 FC 30 03 20
1178- A5 11 A5 9C 20 ED FD A2
1180- 00 60 3B A5 FB F0 1B A5
1188- FA E5 FB 85 FA B0 F8 3B
1190- A9 00 E5 FA 85 FA A2 20
1198- A5 11 CA D0 FA A6 FA 20
11A0- 4A F9 4C 40 11 E6 FE A5
11A8- FE CD 12 0B D0 D3 4C 3A
11B0- FF 3B A5 82 ED 0B 0B 85
11B8- 84 A5 83 ED 0A 0B 85 85
11C0- A9 A4 20 33 11 20 2F 0B
11C8- A5 85 F0 03 20 DA FD A5
11D0- 84 20 DA FD 4C 8E FD A1
11D8- 8B F0 0A 20 ED FD 20 70
11E0- 09 A2 00 F0 F2 60 20 39
11E8- 09 A1 86 C9 9B F0 F6 20
11F0- ED FD A2 00 F0 F0 A9 DB
11F8- 85 8B A9 12 85 89 20 D7
1200- 11 20 E6 11 A9 E2 85 88
1208- A9 12 85 89 20 D7 11 20
1210- 56 12 A9 AC 20 ED FD A9
1218- CC 20 ED FD 4C B1 11 A9
1220- E6 85 8B A9 12 85 89 20
1228- D7 11 20 E6 11 A9 E2 85
1230- 8B A9 12 85 89 20 D7 11
1238- AD 0A 0B 20 DA FD AD 0B
1240- 0B 20 DA FD 20 8E FD 4C
1248- 7F 11 20 ED FD 20 E6 11
1250- 20 8E FD 4C 7F 11 A9 E2
1258- 85 8B A9 12 85 89 20 E6
1260- 11 AD 0A 0B 20 DA FD AD
1268- 0B 0B 4C DA FD E6 9E A0
1270- 9E 20 5F 0A 0B 0B 89 ED
1278- 12 20 7C 11 8B 10 F7 A5
1280- 85 D0 0B A5 84 C9 64 B0
1288- 05 A2 02 20 4A F9 A0 84
1290- 20 5F 0A 0A 06 89 F6 12
1298- 20 7C 11 8B 10 F7 4C 8E
12A0- FD 2A 2A 2A 20 45 52 4F
12A8- 43 20 2A 2A 2A AA AA A0
12B0- C4 C5 D4 AD C5 CC D0 D0
12B8- C1 A0 AA AA 21 4E 45 44
12C0- 4E 55 46 45 47 20 54 48
12C8- 43 49 4E 21 45 42 41 47
12D0- 4E 49 45 20 45 48 43 53
12D8- 4C 41 46 84 C2 D3 C1 D6
12E0- C5 00 AC C1 A4 00 84 C2
12E8- CC CF C1 C4 00 A0 BA C5
12F0- D4 C9 C5 D3 A0 AE CE C5
12F8- CC C9 C5 DA A0 DA A0 00

```

```

10 S = 0
20 FOR I = 0 TO 2815
30 X = PEEK (20000 + I)
40 S = S + X
60 NEXT I
70 IF S = 321673 THEN PRINT "PRUEFSUMME OK"

```

Bild 2. Dieses Programm ermittelt eine Prüfsumme über alle Bytes von Apple-Ted. Der Editor muß vor dem Starten der Prüfroutine ab Adresse 20 000 im Speicher stehen (BLOAD TED, A20 000)

So wird Apple-Ted bedient

Die Anzahl n wird immer dezimal angegeben ($-10\,000 < n < 10\,000$). Läßt man sie weg, so entspricht das $n = 0$. Ein Text folgt unmittelbar auf ein Kommando, bei der Eingabe mit I z. B.: I Dies ist ein Test \$\$.

Folgende Kommandos wirken nur bei der Eingabe:

Escape	Trenner zwischen Befehlen und Texten (Escape wird invers als \$ dargestellt) Abarbeiten des Eingabespeichers nach \$\$
Backspace	(Pfeil rückwärts) Letztes Zeichen löschen und wieder ausgeben
CTRL-R	Letzte Zeile im Eingangsspeicher ausgeben
CTRL-X	Gesamte Eingabe löschen
@x	Delimiter umdefinieren

Die folgenden Kommandos können beliebig gemischt aneinandergehängt werden und werden erst nach Abschluß mit \$\$ ausgeführt (außer CTRL-D-Kommando LO und SA):

nBC	n Zeilen vom Zeiger an im Hilfspeicher an schon Vorhandenes anhängen
nBT	wie nBC, aber Zeilen im Hauptspeicher löschen
BK	Hilfsspeicher löschen
BR	Hilfsspeicherinhalt ausgeben
C	Zeichenkette ändern
nD	n Zeichen löschen
E	Textende im Hauptspeicher (hexadezimal) ausgeben
F	Zeichenkette suchen
H	Rückkehr ins Basic
CTRL-I	Tabulieren (beinhaltet I am Anfang eines Textes)
I	Text einfügen
I CTRL-B	Inhalt des Hilfsspeichers einfügen
J	Zeiger an den Anfang des Textes
nK	n Zeilen löschen ($n = \#$: alles löschen)
OK	vom Zeiger bis zum Anfang der Zeile löschen
LO	Text von Floppy laden (nur Einzelkommando)
nL	Zeiger um n Zeilen versetzen
OL	Zeiger an den Anfang der Zeile
nM	Zeiger um n Zeichen versetzen
N	Länge des Textes (hexadezimal) ausgeben
nPR#k	n Zeilen auf Slot k geben ($0 \leq k < 6$)
SA	Gesamten Text auf Floppy retten (wie LO)
T	Zeigerstellung ausgeben
nT	n Zeilen ausgeben ($n = \#$: alles ausgeben)
V	Zahl der Zeilen pro Seite ausgeben
nX	Die folgenden Befehle n-mal ausführen
nY	Anzahl der Zwischenräume pro TAB definieren
Z	Zeiger an das Ende des Textes
CTRL-D	Zur Benutzung vor DOS-Kommandos

Zeilen werden mit Return abgeschlossen. Return wird als Zeichen wie jedes andere behandelt.

Fehlermeldungen: „NICHT GEFUNDEN“ bei F und C; „*** CORE ***“, wenn kein Speicherplatz mehr frei ist; „FALSCH EINGABE“ bei falscher Behandlung von PR und a. Der Zeiger steht danach am Textanfang. Bei Überlaufen des Eingabespeichers Abbruch des Eingabemodes und Abarbeiten des Eingabespeichers.

Änderungen und Erweiterungen gegenüber der Urversion

Damit man mit einer 40-Zeichen-Anzeige auf dem Bildschirm auch die volle Zeilenlänge eines Druckers bearbeiten kann, ertönt nach der Eingabe von 70 Zeichen ein Piepston. Die relevante Anzahl der Zeichen für diese Klingel steht in Speicherplatz \$812 und kann nach eigenem Geschmack geändert werden. N gibt nicht mehr die Anzahl der Zeichen im Hauptspeicher, sondern die (hexadezimale) Länge des Textes aus, was für die später angesprochenen DOS-Kommandos wichtig ist.

V zeigt nicht mehr die Gesamtzahl der Zeilen des Textes, sondern die Anzahl der Zeilen pro Seite. Als Seite ist definiert, was zwischen dem Anfang des Hauptspeichers und einem Formfeed (CTRL-L), zwei Formfeeds oder einem Formfeed und dem Ende des Hauptspeichers steht. Damit ist ein bequemes Pagenieren gewährleistet.

S(earch) wurde in F(ind) umdefiniert, weil das S für DOS-Befehle benutzt werden sollte.

CTRL-I dient zum Tabulieren und ersetzt außerdem am Anfang eines Textes das I(nsert). Man kann also, wenn man mit linkem Heftrand schreiben will, einen Schreibvorgang mit CTRL-I einleiten. Der Tabulator ist grundsätzlich für den gesamten Text gleich und ist nach dem Start auf das Rastermaß acht eingestellt.

Mit nY kann der Tabulator zwischen 0 und 255 verändert werden. Da er nur bei der Ausgabe wirkt und bei der Eingabe als \$89 im Text weggespeichert wird, geht das auch noch bei fertigen Texten.

Mit 0Y hat man überhaupt keine Tabulierung mehr. Lassen Sie sich dadurch nicht täuschen! Die Tabulatoranweisung ist nach wie vor im Text vorhanden. Bei den Befehlen M und D wird der TAB wie ein Zeichen behandelt. Die Klingel behandelt den TAB wie die gesetzte Anzahl von Zwischenräumen.

nPR#k dient zur Ausgabe auf einen Slot, entspricht also nT, nur daß die Ausgabe nach \$Ck00 umdirigiert wird, wo dann natürlich ein entsprechendes Treiberprogramm stehen muß, damit sich das System nicht aufhängt. nPR#0 wirkt immer auf den Bildschirm, nur daß die Steuerzeichen jetzt verborgen sind.

#PR#k gibt den gesamten Text auf Slot k. nPR#k-Befehle für $k < 0$ oder $k > 5$ sind verboten und werden nicht akzeptiert.

Sollte sich das System wider Erwarten einmal nicht wieder mit dem „Prompt“ melden (der von ! auf ? umdefiniert wur-

de, um Konflikte mit Assemblern zu vermeiden), dann drücken Sie die Reset-Taste. Keine Angst, es geht kein Text verloren, nur der laufende Zeiger steht jetzt wieder am Anfang des Textes. Allerdings: Bei langen Schleifen im Insert-Mode kann man sich eine Menge „Müll“ einfangen, wenn man es nicht abwarten konnte und die Reset-Taste drückte, wenn Ted gerade Platz freigeschoben hat und dann zum Abbruch gezwungen wurde.

Sehr ärgerlich war bei dem alten Editor, daß man Escape, CTRL-X und CTRL-R nicht in den Text einfügen konnte, da das Eingabekommandos sind. Viele Drucker werden durch Escape-Sequenzen gesteuert. Deshalb gibt es jetzt einen Befehl, um den Delimiter, also das Escape, umzudefinieren. Weil das sehr gefährlich sein kann und man damit beliebig viel ungewollten Unfug anrichten kann, sollte man ihn nur sehr vorsichtig und überlegt benutzen. Er gilt deshalb auch nur für das I(nsert)-Kommando und wird nach Abarbeiten des Eingangsspeichers sofort wieder abgeschaltet. Für dieses Umdefinieren wurde der Master-Space (@) verwendet, der leger, aber treffend im Deutschen auch ‚Klammeraffe‘ genannt wird. Das unmittelbar auf den Klammeraffen folgende Zeichen wird als neuer Delimiter für den I(nsert)-Befehl definiert und steht als Textzeichen dann nicht mehr zur Verfügung. Hier ein Beispiel, um den Epson-MX-80 auf Fettdruck umzuschalten:

@.I\$EText.\$\$

\$E steht jetzt mit im Text und schaltet den MX-80 auf Fettdruck um; der Punkt schließt den Text ab, \$\$ den Eingabemodus. Man darf nur ein Escape an einer Stelle einfügen, da für den Eingangsspeicher das Escape immer noch Trennzeichen ist. Für das I ist jetzt aber der Punkt das Trennzeichen; es könnte auch jedes andere Zeichen sein. Nach @ „Char.“ sind CTRL-X, CTRL-R und Backspace in ihrer Funktion stillgelegt, so daß sie ebenfalls Bestandteil des Textes werden können. Es gibt dann keine Korrekturmöglichkeit mehr bei der Eingabe! Zum Glück bleibt immer noch die Reset-Taste, wenn man sich verhaspelt. Wie bei allen komplizierten Sachverhalten gilt auch hier: Probieren geht über Lesen!

Zwei Befehle wurden hinzugefügt, um die mit Apple-Ted erzeugten Texte via DOS zu speichern und zu laden: SA 'Filename' (,Device) speichert zuvor erstellten Text unter dem Namen 'Filename' als Binärfile auf der Floppy, wo-

bei dem Benutzer gleichzeitig Anfang und Länge des Textes mitgeteilt werden.

Beispiel: SA BRIEF, D2\$\$
gibt die Meldung BSAVE BRIEF, D2, A\$2000,L\$xxxx auf den Bildschirm und speichert den erstellten Text auf dem Laufwerk 2 unter dem Namen BRIEF. LO 'Filename' (,Device) lädt den Text wieder von der Floppy in den Hauptspeicher.

Beispiel: LO BRIEF, D2\$\$,
gibt die Meldung BLOAD BRIEF, D2,A\$2000 auf den Bildschirm und lädt den Text BRIEF. Alte Texte werden gelöscht.

Beachten Sie bitte, daß beide Befehle nicht mit CR abgeschlossen werden, sondern wie alle Befehle des TED mit Escape! Beide Befehle können zwar in einen Befehlsstring eingebunden werden, aber alle Befehle nach SA und LO werden nicht mehr ausgeführt, da TED zu Verwaltungszwecken nach diesen Befehlen einen Warmstart machen muß, um die Zeiger neu zu setzen. Der laufende Textzeiger steht danach immer am Anfang des Textes.

Schließlich kann man durch Vorsetzen eines CTRL-D alle DOS-Befehle benutzen, die es gibt. Auch diese Befehle dürfen nicht mit CR, sondern müssen mit \$\$ (zweimal Escape) abgeschlossen werden.

Beispiel: CTRL-D CATALOG, D1\$\$
wirkt in bekannter Weise.

Dadurch hat man die Möglichkeit, Texte beliebig zusammenzubinden. Hier zwei Beispiele:

1. Anhängen von TEXT2 an TEXT1:
LO TEXT1\$\$
mit Echo BLOAD TEXT1, A\$2000
Jetzt mit E\$\$ das Ende des Textes im Hauptspeicher bestimmen, z. B. mit Echo \$34FA
und hierhin den Start von TEXT2 legen mit CTRL-D BLOAD TEXT 2, A\$34FA\$\$
TEXT2 ist damit an TEXT1 angehängt.

2. Einfügen von TEXT2 an beliebiger Stelle von TEXT1: TEXT1 wie oben laden

Mit BK\$CTRL-D BLOAD
TEXT2,A\$1300\$\$

Hilfsspeicher löschen und TEXT2 dort hin laden (TEXT2 darf nicht länger als \$1AFF-\$1300=\$7FF Zeichen lang sein

entsprechend der zur Verfügung stehenden Länge des Hilfsspeichers). Mit BR\$\$ kann man sich vom Vorhandensein von TEXT2 im Hilfsspeicher überzeugen und mit ICTRL-B\$\$ kann man jetzt TEXT2 vor dem laufenden Zeiger in TEXT1 einfügen, was ja durch entsprechendes Positionieren des laufenden Zeigers an beliebiger Stelle geht. Weitere Anwendungen mögen dem Spielbetrieb des geneigten Lesers überlassen bleiben.

Bei einer falschen Eingabe nach CTRL-D erhält man eine DOS-Basic-Fehlermeldung und befindet sich anschließend im Basic. Ein Reset bewirkt den Rücksprung zum Texteditor.

Auch mit 80-Zeichen-Karte funktionsfähig

Das Programm läuft ohne Änderung auf dem Basis 108 im 40-Zeichen-Modus, wenn man mit FP40 „hochzapt“. Bei einem Start mit FP80, also im 80-Zeichen-Modus, sollte vorher im Programm der Speicherplatz \$825 von 0 auf \$40 geändert werden.

In beschränkter Weise läuft Apple-Ted auch mit einer 80-Zeichen-Karte auf dem Apple (z. B. Videx). Steckt diese Karte in Slot 3, reicht es, im Programm \$825 von 0 auf \$80 und \$B30, \$B31 von \$58, \$FC auf \$ED, \$FD zu ändern. Anderenfalls muß noch \$B7D von \$C3 auf \$Cx (x = Slotnummer) geändert werden. Die 80-Zeichen-Karte muß eingeschaltet werden, bevor man mit BRUN TED in den Editor geht, und nach dem Start von Apple-Ted muß einmal Reset gedrückt werden. Da das Programm sich sonst aufhängt, wird Ihnen gar nichts anderes übrigbleiben! Sonderfunktionen der 80-Zeichen-Karte stehen allerdings nicht mehr zur Verfügung, zum Beispiel die Kleinschreibungskrücke mit CTRL-A (es gibt inzwischen viel elegantere Methoden, den Apple auf Kleinschreibung umzurüsten).

Bild 2 zeigt ein kleines Prüfsummenprogramm für die in Bild 1 wiedergegebene Version. Bevor es gestartet wird, ist Apple-Ted ab Adresse A 20 000 (dez.) zu laden (BLOAD TED, A 20 000).

Literatur

- [1] Edwards, Lew: The First Book of KIM, MOVIT.
- [2] Ihme, H. M.: 6502-Texteditor. Funkschau-Sonderheft Nr. 31, Seite 49...52.
- [3] Apple II Reference Manual.
- [4] Beneath Apple DOS.

Herwig Feichtinger

Datenaustausch

Zwei Computer im Gespräch

Wer zwei unterschiedliche Computer nebeneinander stehen hat, ärgert sich oft darüber, daß weder Kassetten- noch Diskettenformat kompatibel sind. Ein Überspielen von Programmen und Daten ist daher nur per Kabel möglich. Doch auch dafür braucht man schon ein wenig Software. Hier wird ein Apple-II-Programmpaket vorgestellt, das eine Übertragung im 6502-Hex-Format erlaubt. Als Gegenstation dient beispielhaft ein AIM-65.

Das 6502-Hex-Format ist ein Format zur Datenübermittlung, das beliebige Speicherbereiche in ASCII-Hexadezimal-Ziffern verschlüsselt und mit Prüfsummen gegen Fehler absichert. Zahlreiche Computer haben dafür geeignete Routinen bereits in ihrem Monitorprogramm implementiert, z. B. KIM-1, AIM-65 oder System-65. Das Format ist folgendermaßen aufgebaut: Jede Zeile beginnt mit einem Strichpunkt und endet mit einem Return-Zeichen. Nach dem Strichpunkt folgen zwei Hex-Ziffern, die die Anzahl der Datenbytes der Zeile angeben. Dann kommt die vierstellige Hexadezimal-Anfangsadresse (MSB, LSB), gefolgt von der spezifizierten Anzahl von Datenbytes. Am Ende der Zeile steht eine vierstellige hexadezimale Prüfsumme über alle vorangegangenen Bytes (außer dem Strichpunkt; MSB, LSB). Die maximale Datenbyte-Anzahl einer Zeile beträgt dezimal 24 (hex 18). Ist die Zahl der Datenbytes in einer Zeile mit Null spezifiziert, so handelt es sich um die letzte Zeile der Übertragung; statt der Adresse wird dann vierstellig die Zahl der vorangegangenen Zeilen und wieder die Prüfsumme gesendet, wobei letztere natür-

lich mit der Zeilenzahl identisch ist, weil ja keine weiteren Bytes in der Schlußzeile stehen.

Drei Drähte genügen

Der Einfachheit halber erfolgt die Übertragung zwischen Apple-II und AIM-65 asynchron mit 300 Baud im TTL-Pegel. Auf Seiten des Apple-II kann dann nämlich der Game-Connector als Schnittstelle mißbraucht werden, und der AIM-65 ist mit wenigen Änderungen an der TTY-Schnittstelle ebenfalls hierfür geeignet (Bild 1). Insgesamt sind also nur drei Drähte zwischen beiden Computern erforderlich.

Bild 2 zeigt das Assemblerlisting des Programms, das den Apple-II veranlaßt, Daten im 6502-Hex-Format zu senden oder zu empfangen. Will man Daten vom AIM-65 empfangen, so braucht man nur im Apple-Monitor 8200G einzugeben und auf Seiten des AIM ein Mini-Programm zu starten (Bild 3). Nach Druck auf die AIM-Taste D (Dump) und Beantworten der Fragen FROM und TO mit Anfangs- und Endadresse gibt man OUT = U ein. Ist der vollständige Adres-

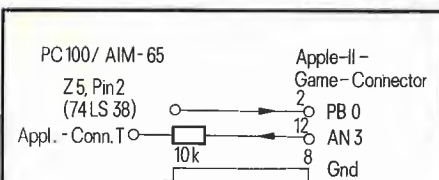


Bild 1. Verbindung eines AIM-65 mit dem Apple-II. Die Übertragung erfolgt asynchron im TTL-Pegel; auf der AIM-Platine sind C7 und R8 auszulöten

0800	5	ERROR	EQU	\$FF2D	823E	A8	48	TAY
0800	6	AN3	EQU	\$C05E	823F	995EC0	49	STA AN3,Y
0800	7	PB	EQU	\$C061	8242	A4C9	50	LDY XTEMP
0800	8	XTEMP	EPZ	\$C9	8244	68	51	PLA
0800	9	XTEMP1	EPZ	\$CA	8245	60	52	RET RTS
0800	10	PNT	EPZ	\$CB	8246		53	;
0800	11	SUM	EPZ	\$CD	8246		54	; ZEICHENEINGABE
0800	12	LEN	EPZ	\$CF	8246	86CA	55	ZEIN STX XTEMP1
0800	13	CNT	EPZ	\$D1	8248	2C61C0	56	EMP BIT PB
0800	14				824B	30FB	57	BMI EMP
8200	15		ORG	\$8200	824D	A208	58	LDX #8
8200	4CDC82	16	JMP	READ	824F	A911	59	LDA #\$11
8203	4C2183	17	JMP	WRITE	8251	203682	60	JSR WAIT+2
8206		18			8254	2C61C0	61	BIT PB
8206		19			8257	30EF	62	BMI EMP
8206	48	20	ZAUS	PHA	8259	A900	63	LDA #0
8207	48	21		PHA	825B	48	64	NBIT PHA
8208	86CA	22	STX	XTEMP1	825C	203482	65	JSR WAIT
820A	A208	23	LDX	#8	825F	AD61C0	66	LDA PB
820C	203482	24	BEGA	JSR WAIT	8262	0A	67	ASL
820F	A900	25		LDA #0	8263	68	68	PLA
8211	8D5EC0	26		STA AN3	8264	6A	69	ROR
8214	68	27		PLA	8265	CA	70	DEX
8215	48	28	WIEDH	PHA	8266	D0F3	71	BNE NBIT
8216	203482	29		JSR WAIT	8268	48	72	PHA
8219	68	30		PLA	8269	203482	73	JSR WAIT
821A	203982	31		JSR OUT2	826C	68	74	PLA
821D	4A	32		LSR	826D	A6CA	75	LDX XTEMP1
821E	CA	33		DEX	826F	0980	76	ORA #\$80
821F	D0F4	34		BNE WIEDH	8271	60	77	RTS
8221	203482	35		JSR WAIT	8272		78	;
8224	A901	36		LDA #1	8272		79	; SUMME AUFADDIEREN
8226	203982	37		JSR OUT2	8272		80	; UND BYTE AUSGEBEN
8229	A9A0	38		LDA #\$A0	8272	20CE82	81	NADD JSR ADD+3
822B	203682	39		JSR WAIT+2	8275		82	;
822E	A6CA	40		LDX XTEMP1	8275		83	; BYTE AUSGEBEN
8230	68	41		PLA	8275	48	84	NUMA PHA
8231	4CF0FD	42		JMP \$FDF0	8276	4A	85	LSR
8234	A922	43	WAIT	LDA #\$22	8277	4A	86	LSR
8236	4CA8FC	44		JMP \$FCA8	8278	4A	87	LSR
8239	48	45	OUT2	PHA	8279	4A	88	LSR
823A	84C9	46		STY XTEMP	827A	208082	89	JSR NOUT
823C	2901	47		AND #1	827D	68	90	PLA

Bild 2. Assemblerlisting des Apple-Programms zum Datenaustausch im 6502-Hex-Format

827E	290F	91		AND	#\$F	8305	91CB	174		STA	(PNT),Y
8280	18	92	NOUT	CLC		8307	20BD82	175		JSR	INCP
8281	69B0	93		ADC	#\$B0	830A	CA	176		DEX	
8283	C9BA	94		CMP	#\$BA	830B	D0F1	177		BNE	READ1
8285	9002	95		BCC	LT10	830D	208C82	178		JSR	BYTIN
8287	6906	96		ADC	#6	8310	C5CE	179		CMP	SUM+1
8289	4C0682	97	LT10	JMP	ZAUS	8312	D009	180		BNE	ERR
828C		98				8314	208C82	181		JSR	BYTIN
828C		99		;	BYTE EINLESEN	8317	C5CD	182		CMP	SUM
828C		100		;	C=1 WENN NICHT HEX	8319	D002	183		BNE	ERR
828C	204682	101	BYTIN	JSR	ZEIN	831B	F0BF	184		BEQ	READ
828F	20A882	102		JSR	PACK	831D	202DFF	185	ERR	JSR	ERROR
8292	B012	103		BCS	NOHEX	8320	60	186	RDY	RTS	
8294	0A	104		ASL		8321		187			
8295	0A	105		ASL		8321		188		;	HEXDATEI AUSGEBEN
8296	0A	106		ASL		8321	AD72AA	189	WRITE	LDA	\$AA72
8297	0A	107		ASL		8324	85CB	190		STA	PNT
8298	85C9	108		STA	XTEMP	8326	AD73AA	191		LDA	\$AA73
829A	204682	109		JSR	ZEIN	8329	85CC	192		STA	PNT+1
829D	20A882	110		JSR	PACK	832B	AD60AA	193		LDA	\$AA60
82A0	B004	111		BCS	NOHEX	832E	85CF	194		STA	LEN
82A2	05C9	112		ORA	XTEMP	8330	AD61AA	195		LDA	\$AA61
82A4	18	113		CLC		8333	85D0	196		STA	LEN+1
82A5	60	114		RTS		8335	A900	197		LDA	#0
82A6	38	115	NOHEX	SEC		8337	85D1	198		STA	CNT
82A7	60	116		RTS		8339	85D2	199		STA	CNT+1
82A8	C9B0	117	PACK	CMP	#\$B0	833B	A98D	200	WR1	LDA	#\$8D
82AA	90FA	118		BCC	NOHEX	833D	200682	201		JSR	ZAUS
82AC	C9C7	119		CMP	#\$C7	8340	A9BB	202		LDA	"
82AE	B0F6	120		BCS	NOHEX	8342	200682	203		JSR	ZAUS
82B0	C9BA	121		CMP	#\$BA	8345	E6D1	204		INC	CNT
82B2	9006	122		BCC	PAK1	8347	D002	205		BNE	WR0
82B4	C9C0	123		CMP	#\$C0	8349	E6D2	206		INC	CNT+1
82B6	90EE	124		BCC	NOHEX	834B	20C482	207	WR0	JSR	CLSUM
82B8	6908	125		ADC	#8	834E	A5D0	208		LDA	LEN+1
82BA	290F	126	PAK1	AND	#\$F	8350	D008	209		BNE	WR2
82BC	60	127		RTS		8352	A5CF	210		LDA	LEN
82BD		128				8354	F039	211		BEQ	WRDY
82BD		129		;	POINTER INKREMENTIEREN	8356	C918	212		CMP	#24
82BD	E6CB	130	INCP	INC	PNT	8358	9002	213		BCC	WR3
82BF	D002	131		BNE	INCP1	835A	A918	214	WR2	LDA	#24
82C1	E6CC	132		INC	PNT+1	835C	AA	215	WR3	TAX	
82C3	60	133	INCP1	RTS		835D	207282	216		JSR	NADD
82C4		134				8360	A5CC	217		LDA	PNT+1
82C4		135		;	CHECKSUMME LOESCHEN	8362	207282	218		JSR	NADD
82C4	A900	136		CLSUM	LDA #0	8365	A5CB	219		LDA	PNT
82C6	85CD	137			STA SUM	8367	207282	220		JSR	NADD
82C8	85CE	138			STA SUM+1	836A	A000	221	WR4	LDY	#0
82CA	60	139			RTS	836C	B1CB	222		LDA	(PNT),Y
82CB		140				836E	207282	223		JSR	NADD
82CB		141		;	BYTE LESEN UND	8371	20BD82	224		JSR	INCP
82CB		142		;	SUMME AUFADDIEREN	8374	A5CF	225		LDA	LEN
82CB	208C82	143	ADD	JSR	BYTIN	8376	38	226		SEC	
82CE	08	144		PHP		8377	E901	227		SBC	#1
82CF	48	145		PHA		8379	85CF	228		STA	LEN
82D0	18	146		CLC		837B	B002	229		BCS	WR5
82D1	65CD	147		ADC	SUM	837D	C6D0	230		DEC	LEN+1
82D3	85CD	148		STA	SUM	837F	CA	231	WR5	DEX	
82D5	9002	149		BCC	ADD1	8380	D0E8	232		BNE	WR4
82D7	E6CE	150		INC	SUM+1	8382	A5CE	233		LDA	SUM+1
82D9	68	151	ADD1	PLA		8384	207582	234		JSR	NUMA
82DA	28	152		PLP		8387	A5CD	235		LDA	SUM
82DB	60	153		RTS		8389	207582	236		JSR	NUMA
82DC		154				838C	4C3B83	237		JMP	WR1
82DC		155		;	HEXDATEN EINLESEN	838F	A900	238	WRDY	LDA	#0
82DC	204682	156	READ	JSR	ZEIN	8391	207582	239		JSR	NUMA
82DF	C9BB	157		CMP	"	8394	A5D2	240		LDA	CNT+1
82E1	D0F9	158		BNE	READ	8396	207582	241		JSR	NUMA
82E3	20C482	159		JSR	CLSUM	8399	A5D1	242		LDA	CNT
82E6	20CB82	160		JSR	ADD	839B	207582	243		JSR	NUMA
82E9	B032	161		BCS	ERR	839E	A5D2	244		LDA	CNT+1
82EB	C900	162		CMP	#0	83A0	207582	245		JSR	NUMA
82ED	F031	163		BEQ	RDY	83A3	A5D1	246		LDA	CNT
82EF	AA	164		TAX		83A5	207582	247		JSR	NUMA
82F0	20CB82	165		JSR	ADD	83A8	A203	248		LDX	#3
82F3	B028	166		BCS	ERR	83AA	A900	249	FILL	LDA	#0
82F5	85CC	167		STA	PNT+1	83AC	207582	250		JSR	NUMA
82F7	20CB82	168		JSR	ADD	83AF	CA	251		DEX	
82FA	B021	169		BCS	ERR	83B0	D0F8	252		BNE	FILL
82FC	85CB	170		STA	PNT	83B2	A98D	253		LDA	#\$8D
82FE	20CB82	171	READ1	JSR	ADD	83B4	200682	254		JSR	ZAUS
8301	B01A	172		BCS	ERR	83B7	60	255		RTS	
8303	A000	173		LDY	#0			256		END	

senbereich übertragen, kann man noch einen weiteren senden, der auch an einer ganz anderen Stelle im Speicher stehen darf; andernfalls ist nach MORE? einfach N einzugeben.

Tritt während der Übertragung ein Fehler auf, so bricht der Apple-II das Einlesen der Daten ab, gibt einen kurzen Ton aus und druckt „ERR“ auf den Bildschirm. Ist die Übertragung ohne Fehler zu Ende, so erscheint wieder der normale Monitor-Prompt. Selbstverständlich erscheinen die Daten im Apple-II im gleichen Speicherbereich, aus dem sie im AIM-65 ausgelesen wurden.

Vom Apple zum AIM

Zum Senden eines Speicherbereichs holt sich der Apple-II die Anfangsadresse aus den DOS-Zellen AA72 und AA73 sowie die Länge aus AA60 und AA61. Dorthin werden sie nach einem BLOAD-Befehl nämlich vom Disketten-Betriebssystem (48-KByte-Apple) automatisch geschrieben. Sinnvoll ist also folgende Vorgehensweise: Erst das Programm in Bild 2 laden, dann erst das zu sendende Maschinenprogramm. Die Zeiger sind dann nämlich von selbst richtig gesetzt, und der Start der Übertragung kann mit 8203G vom Monitor aus erfolgen. Vorher ist natürlich der AIM-65 entsprechend vorzubereiten; Bild 3 enthält bereits das Setzen des dafür nötigen User-Input-Vektors. Dann braucht man nur noch L (für Load) und IN=U einzugeben. Selbstverständlich beschwert sich auch das AIM-Monitorprogramm mit einer Fehlermeldung, wenn ein Übertragungsfehler auftritt, wenn auch mangels Lautsprecher nicht akustisch.

```
<*>=108
</> 0108 DB EB F1 00 ;Vektoren
<*>=F0
</> 00F0 60 90 FD 68
</> 00F4 4C A8 EE ;TTY-Ausg.
<*>=A417
</> A417 0C C0 ;300 Baud
```

Bild 3. Vorbereiten des AIM-65 zur seriellen Ein- und Ausgabe

Baudrate	1200	300
Apple	\$8235 \$8250	\$0F \$22 \$0A \$15
AIM	\$A417 \$A418	\$02 \$0C \$FD \$C0

Bild 4. Baudraten-Einstellung bei AIM-65 und Apple-II

Universell verwendbar

Systemspezifisch sind in Bild 2 lediglich die Adressen zur Zeichen-Ein- und -Ausgabe sowie die Error-Routine im Apple-Monitorprogramm. Paßt man diese an, so läßt sich die Software für beliebige andere 6502-Computer verwenden. Ebenso gut ist es natürlich möglich, zwei Apples damit in Verbindung treten zu lassen.

Wem 300 Bd zu langsam sind, kann die Programmparameter für 1200 Bd ändern (Bild 4). Das Byte an der Apple-Adresse 822A bestimmt die Verzögerungszeit zwischen zwei gesendeten Zeichen; der in Bild 2 angegebene Wert hex A0 gestattet es, auch solche Geräte als Gegensta-

tion zu verwenden, die die empfangenen Zeichen als Echo zurücksenden oder eine längere Zeit zur Verarbeitung einzelner Zeichen brauchen. Im allgemeinen kommt man aber mit einem Wert von hex 15 in Zelle 822A aus.

Das Programm ist als Objektcode und als Assembler-Source für den Lazer-Assembler „Lisa“ vom Franzis-Software-Service auf der Apple-Sammeldiskette 6 erhältlich.

Literatur

- [1] Hofer, R.: V.24-Ein- und Ausgabe beim Apple-II. mc 1983, Heft 3.
- [2] Feichtinger, H.: Babylon's Datenverwirrung. mc 1981, Heft 1.

Groß- und Kleinschrift ohne Shift-Taste

Das im Bild dargestellte Applesoft-Programm hat sich in einer Zahnarztpraxis bei der Erfassung von Adressen bewährt.

Es sorgt dafür, daß der jeweils erste Buchstabe einer Eingabezeile groß geschrieben wird, ohne daß man die Shift-Taste betätigen muß. Außerdem erscheinen diejenigen Buchstaben in Großschrift, denen ein Leerzeichen, ein Punkt oder ein Bindestrich vorangeht.

Das Unterprogramm zur Umwandlung der Zeichen in Kleinbuchstaben besteht nur aus den Zeilen 101 bis 107. Wie es aufgerufen wird, geht aus dem Programmteil ab Zeile 1000 hervor.

Kommas, Doppelpunkte und Strichpunkte ersetzt das Unterprogramm durch ein Leerzeichen. Dadurch verhindert es, daß unbemerkt Teile einer Eingabe verlorengehen. Damit keine Verzögerung bei der Ausgabe auf dem Bildschirm eintritt, sollte die Routine möglichst am Programmanfang stehen. Da jedes neue Wort automatisch groß geschrieben wird, muß man, wenn ein Kleinbuchstabe gewünscht wird, diesen einen Buchstaben mit der Shift-Taste eingeben – z. B. das „d“ bei „Unter den Linden“. Kommen zwei Großbuchstaben hintereinander vor, dann muß man sie durch ein Leerzeichen trennen.

Dr. Wolfgang Zechner

```
10 GOTO 1000
101 IF I$ = "," OR I$ = ";" OR I$ = ":" THEN I$ = " "
103 IF N$ = "" OR RIGHT$(N$,1) = " " OR RIGHT$(N$,1) = "." OR RIGHT$(N$,1) = "-" THEN RETURN
105 I = ASC(I$); IF I < 65 OR I > 93 THEN RETURN
107 I = I + 32; I$ = CHR$(I); RETURN
1000 PRINT "ANREDE,TITEL: ";
1001 GET I$: IF I$ = CHR$(13) THEN PRINT : GOTO 1004
1002 GOSUB 101: PRINT I$;
1003 N$ = N$ + I$: GOTO 1001
1004 TI$ = N$: N$ = ""
1005 PRINT "VORNAME,NAME: ";
1006 GET I$: IF I$ = CHR$(13) THEN PRINT : GOTO 1009
1007 GOSUB 101: PRINT I$;
1008 N$ = N$ + I$: GOTO 1006
1009 NN$ = N$: N$ = ""
1010 PRINT "ADRESSE: ";
1011 GET I$: IF I$ = CHR$(13) THEN PRINT : GOTO 1014
1012 GOSUB 101: PRINT I$;
1013 N$ = N$ + I$: GOTO 1011
1014 AD$ = N$: N$ = ""
```

Die Eingabe von Adressen beschleunigt dieses kleine Programm. Es wurde auf einem Apple-kompatiblen Computer mit Groß-/Kleinschreibung erstellt

Ludwig Neidl

Felder schnell abgespeichert

Neuer Apple-Befehl macht's möglich

Haben Sie schon einmal versucht, ein großes Datenfeld – etwa mit 40 x 40 Indizes – unter dem Disketten-Betriebssystem des Apple-II abzuspeichern oder zu lesen? Probieren Sie es aus! Lehnen Sie sich entspannt zurück und schlürfen Sie einen Drink, den ein Mensch mit empfindlichen Nerven bei einer solchen Aktion dringend braucht. Die Standard-DOS-Routinen brauchen dafür eine „halbe Ewigkeit“. Der Grund liegt darin, daß sie die Daten als ASCII-Zeichen auf die Diskette schreiben und auch die internen DOS-Routinen ihre Zeit benötigen. Viel einfacher ist es, das Feld direkt binär abzuspeichern.

Das abgedruckte Programm (siehe Bild) speichert und liest Felder von Gleitkomma- und Ganzzahlen. Es wird in den Bereich \$2F5 bis \$3CB geschrieben und mit 2F5G initialisiert. Sollte jemand danach den gesamten Eingabepuffer (\$200 bis \$2FF) vollschreiben, so wird zwar der Initialisierungsteil zerstört, die Haupt-routine bleibt jedoch erhalten. Mit der Floppy kann man dieses Hilfsprogramm mit BRUN ARRAY COPY initialisieren, sofern es vorher ab Adresse \$2F5 unter dem Namen ARRAY-COPY abgespei-chert wurde. Die Routine kann über den &-Vektor von Applesoft aus aufgerufen werden und funktioniert auch im Direktmodus.

Ein Zahlenfeld wird auf Diskette geschrieben

Aufruf durch: & STORE A
bzw. allgemeiner:
(... :) & STORE A (, expr) (:...)
& STORE A % (, expr)
& STORE A \$ (, expr)

Nach dem Aufruf wird das Zahlenfeld auf die Diskette geschrieben, wobei der Wert (expr) zwischen 0 und 255 liegen muß. Wird kein Feldindex vereinbart, so

nimmt die Routine für den Ausdruck (expr) den Wert 0 an. Das Feld steht nach dem Schreibprozeß mit seinem Feldnamen (A oder A %, aber auch A \$!) und dem Anhang: /(expr) auf der Diskette, also z. B. A/00 beim Aufruf: & STORE A.

Die Routine kann auch die Stringpointer eines Stringfeldes (& STORE A \$) abspeichern. Eine direkte Anwendungsmöglichkeit dieser Eigenschaft läßt sich zwar nicht erkennen, die Möglichkeit hierzu sollte jedoch dem Anwender nicht ver-schlossen bleiben.

Und so liest man das Feld

Aufruf über:
(...:) & RECALL A (, expr) (:...)
(...:) & RECALL A % (, expr) (:...)
(...:) & RECALL A \$ (, expr) (:...)

Hierbei wird das Feld mit dem Namen A (bzw. A % oder A \$) von der Diskette in den Bereich des Feldes im RAM eingelesen. Natürlich muß das Feld vorher vereinbart worden sein (DIM A (...)).

Das Feld im RAM muß nicht die gleiche Dimensionierung aufweisen, jedoch

mindestens die gleiche Gesamtlänge wie das Feld auf der Diskette besitzen. In einem solchen Fall stimmt die Indizierung natürlich nicht mehr überein (siehe dazu das „Apple Reference Manual“, S. 137). Ist das Feld, das eingelesen wird, größer als der vereinbarte Bereich im RAM, so muß man mit einem Absturz rechnen, da dann die „Header“ anderer Felder, die im RAM auf das eingelesene Feld folgen, zerstört werden.

Die Vorteile dieser Methode seien anhand der Tabelle 1 und 2 dargestellt. Sie zeigen die Bearbeitungszeiten und die benötigten Speicherkapazitäten für große Datenfelder. Hierbei ist die kon-ventionelle Methode mit (sequentiellen) Textfiles stark vom jeweiligen Argument abhängig (die direkte Methode mit & STORE ist vom Inhalt der Felder völlig unabhängig). Die jeweils zweite Zeile zeigt die benötigten Werte für Datenfel-der, in denen nur Nullen stehen. Im Regelfall wird man jedoch pro Zahl 14...15 Gleitkomma bzw. 5...6 (Integer) ASCII-Zeichen benötigen, so daß die Angaben der dritten Zeile durchaus häufiger vor-kommen können.

Der Vergleich der Methoden ergibt für & STORE eine 7- bis 8mal höhere Ab-speichergeschwindigkeit sowie die Re-duzierung des Speicherbedarfes auf bis zu 30 % des ursprünglichen Wertes.

Tabelle 1: Abspeichern des Gleitkommazahlen-Feldes A (40,40)

Methode	Argument	Zeit(s)	Blöcke
& STORE	„beliebig“ (5 Bytes)	12	34
Textfile	0 (2 Bytes)	38	15
Textfile	-0.3333333E-02 (16 Bytes)	104	107

Tabelle 2: Abspeichern des Ganzzahlen-Feldes A % (40,40)

Methode	Argument	Zeit(s)	Blöcke
& STORE	„beliebig“ (2 Bytes)	7	15
Textfile	0 (2 Bytes)	38	15
Textfile	-12345 (7 Bytes)	60	47

[illegible]

Zusätzliche Befehle zum Abspeichern und Laden von Feldern stellt dieses Programm zur Verfügung. Bei größeren Datenmengen spart das Diskettenplatz und lästige Wartezeiten

```

037B 8A      110      TXA                      ; TEST AUF % (-) $
037C 1004    111      BPL STRING
037E A9A5    112      LDA #"%"
0380 D002    113      BNE INTSTR
0382 A9A4    114      STRING LDA #" $"
0384 20EDFD  115      INTSTR JSR COUT
0387 A9AF    116      FLOAT LDA #" /"
0389 20EDFD  117      JSR COUT                      ; "/" DRUCKEN
038C A542    118      LDA NUMBER
038E 20DAFD  119      JSR PRBYTE                      ; ARRAYINDEX DRUCKEN
0391 A9C4    120      LDA #BGNSTR
0393 A003    121      LDY /BGNSTR
0395 203ADB  122      JSR STROUT                      ; (,A$) DRUCKEN
0398 A63C    123      LDX ALL
039A A53D    124      LDA A1H
039C 2041F9  125      JSR PRNTAX                      ; ANFANGSADRESSE IN HEX DRUCKEN
039F 28      126      PLP                          ; L/S-FLAG HOLEN
03A0 B00E    127      BCS NOSAVE                      ; BEI "LOAD" SPRINGEN
03A2 A9C8    128      LDA #LENSTR
03A4 A003    129      LDY /LENSTR
03A6 203ADB  130      JSR STROUT                      ; (,L$) DRUCKEN
03A9 A63E    131      LDX A2L
03AB A53F    132      LDA A2H
03AD 2041F9  133      JSR PRNTAX                      ; LAENGE IN HEX DRUCKEN
03B0 208EFD  134      NOSAVE JSR CROUT                  ; CRLF ALS ENDE DES DOS-BEFEHLS
03B3 60      135      RTS
03B4 84      136      LOADST HEX 84                      ; CTRL-D
03B5 C2CCCF  137      ASC "BLOAD "
03B8 C1C4A0
03BB 00      138      HEX 00
03BC 84      139      SAVEST HEX 84                      ; CTRL-D
03BD C2D3C1  140      ASC "BSAVE "
03C0 D6C5A0
03C3 00      141      HEX 00
03C4 ACC1A4  142      BGNSTR ASC ",A$"
03C7 00      143      HEX 00
03C8 ACCCA4  144      LENSTR ASC ",L$"
03CB 00      145      HEX 00
                146      END

```

***** END OF ASSEMBLY

```

*****
*
* SYMBOL TABLE -- V 1.5 *
*
*****

```

LABEL. LOC. LABEL. LOC. LABEL. LOC.

** ZERO PAGE VARIABLES:

```

A1L  003C  A1H  003D  A2L  003E  A2H  003F  A3L  0040  A3H  0041
A4L  0042  NUMBER 0042  TXTPTR 00B8  LOWTR 009B

```

** ABSOLUTE VARIABLES/LABELS

```

CHRGET 00B1  CHRGOT 00B7
AMPRSN 03F5  STROUT DB3A  GETBYT E6F8  GTARPT F7D9  PRNTAX F941  COUT  FDED
CROUT  FD8E  PRBYTE FDDA  START 0300  SAVE 030A  NOERR 0311  ALLRGT 0312
NOARG 032E  DOIT 035E  NOSCBU 0378  STRING 0382  INTSTR 0384  FLOAT 0387
NOSAVE 03B0  LOADST 03B4  SAVEST 03BC  BGNSTR 03C4  LENSTR 03C8

```

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:012A

!
BRUN SORT.CODE
SYMBOL TABLE SORTED ALPHABETICALLY

```

A1H  003D  A1L  003C  A2H  003F  A2L  003E  A3H  0041  A3L  0040
A4L  0042  ALLRGT 0312  AMPRSN 03F5  BGNSTR 03C4  CHRGET 00B1  CHRGOT 00B7
COUT  FDED  CROUT  FD8E  DOIT 035E  FLOAT 0387  GETBYT E6F8  GTARPT F7D9
INTSTR 0384  LENSTR 03C8  LOADST 03B4  LOWTR 009B  NOARG 032E  NOERR 0311
NOSAVE 03B0  NOSCBU 0378  NUMBER 0042  PRBYTE FDDA  PRNTAX F941  SAVE 030A
SAVEST 03BC  START 0300  STRING 0382  STROUT DB3A  TXTPTR 00B8

```

SYMBOL TABLE SORTED BY ADDRESS

```

ALL  003C  A1H  003D  A2L  003E  A2H  003F  A3L  0040  A3H  0041
NUMBER 0042  A4L  0042  LOWTR 009B  CHRGET 00B1  CHRGOT 00B7  TXTPTR 00B8
START 0300  SAVE 030A  NOERR 0311  ALLRGT 0312  NOARG 032E  DOIT 035E
NOSCBU 0378  STRING 0382  INTSTR 0384  FLOAT 0387  NOSAVE 03B0  LOADST 03B4
SAVEST 03BC  BGNSTR 03C4  LENSTR 03C8  AMPRSN 03F5  STROUT DB3A  GETBYT E6F8
GTARPT F7D9  PRNTAX F941  CROUT  FD8E  PRBYTE FDDA  COUT  FDED

```


Wolfgang Schöpe

Mehr Platz auf Apple-Disketten

Beim Apple-DOS 3.3 stehen normalerweise 35 Spuren auf der Diskette zur Verfügung, was einem Speicherplatz von 143 KByte entspricht. Davon verbleiben nach Abzug des Betriebssystems selbst knapp 127 KByte für den Benutzer. Durch Ändern weniger Adressen im DOS kann man jedoch 80 Sektoren (20 480 Byte) hinzugewinnen.

Wenige Änderungen im DOS genügen, um statt 35 nun 40 Spuren auf der Diskette auszunützen. Beim 48-KByte-DOS bestimmen die Adressen hex BEFE und AEB5 die Anzahl von Spuren und Sektoren. Die nötigen Anpassungen für 40 Spuren sind also:

Adresse	alt	neu
BEFE	23	28
B3EF	23	28
AEB5	8C	A0

Die Adresse B3EF, die im VTCO-Puffer liegt, gibt die neuen Spuren zur Belegung frei.

Am einfachsten ist es, nach diesen Änderungen mit INIT eine neue Diskette zu initialisieren. Auf ihr steht dann das geänderte DOS für 40 Spuren. Dabei sollte gleich das gewünschte Basic-HELLO-Programm im Speicher stehen; würde man nämlich vor INIT erst eines von

einer anderen Diskette laden, so wäre der VTCO-Puffer wieder auf 35 Spuren zurückgestellt.

Das Umstellen alter, schon beschriebener Disketten auf 40 Spuren ist nicht ganz so einfach. Es müssen nämlich die im DOS liegenden Adressen auf der Disk überschrieben werden, ohne den sonstigen Disketteninhalt zu ändern:

BEFE in Spur 0 auf Sektor 8, Adresse FE; B3EF in Spur 2 auf Sektor 2, Adresse EF; AEB5 in Spur 1 auf Sektor D, Adresse B5; VTCO in Spur 11, Sektor 0, Adresse 34. In der Bit-Kartei des VTCO-Puffers müssen außerdem die neuen Spuren auch freigegeben werden, indem man die Adressen C4, C5, C8, C9, CC, CD, D0, D1, D4 und D5 auf hex FF setzt.

Um diesen Vorgang benutzerfreundlich durchzuführen, übernimmt das Maschinenprogramm in Bild 1 das Lesen und

```
*0300.03BF
0300- 20 E3 03 04 06 05 07 A0
0308- 01 A9 00 91 06 C8 A5 19
0310- 91 06 C8 A9 00 91 06 C8
0318- A5 1A 91 06 C8 A5 18 91
0320- 06 A0 08 A9 00 91 06 C8
0328- A9 10 91 06 A0 0C A5 1C
0330- 91 06 A0 0E A9 00 91 06
0338- C8 A9 00 91 06 A4 06 A5
0340- 07 20 D9 03 A9 00 85 48
0348- A0 0D B1 06 85 1D 60 20
0350- E3 03 04 06 05 07 A0 01
0358- A9 00 91 06 C8 A5 19 91
0360- 06 C8 A9 00 91 06 C8 A5
0368- 1A 91 06 A0 0C 91 06 A4
0370- 06 A5 07 20 D9 03 8D 89
0378- C0 A5 1A 85 44 A9 00 85
0380- 41 A9 A8 85 3E A9 28 85
0388- 45 A9 00 8D CB BE 8D F7
0390- BE A0 56 A9 00 20 BF BE
0398- 20 0D BF A9 00 B0 05 20
03A0- E0 BE 90 06 A0 0D B1 06
03A8- 85 1D A9 A9 8D CB BE A9
03B0- B0 8D F7 BE 8D 88 C0 A9
03B8- 00 85 48 60 00 00 00 00
```

Bild 1. Das Maschinenprogramm. Es enthält die USRWTS-Routine aus mc 6/1983

Zurückschreiben von bzw. auf Diskette, und ein Basic-Programm (Bild 2). Dieses Basic-Programm lädt beim Start zunächst das Maschinenprogramm nach. Dann wird der Benutzer gebeten, die auf 40 Spuren umzustellende Diskette einzulegen. Die neuen fünf Spuren werden initialisiert und stehen dann als zusätzliche 20 KByte zur Verfügung. Die hier abgedruckten Programme sind Bestandteil der Apple-Sammeldiskette 4, die unser Software-Service liefert.

Literatur

- [1] Wiegandt, Dr. Ralf: Apple-DOS – Arbeitsweise und Aufbau. mc 1983, Heft 6, Seite 53.

```
3LIST
100 REM *****
110 REM ***UPDATE VON 35 AUF 40 TRACK *****
120 REM *** W. SCHÖPE / 03.01.1984 *****
130 REM *****
140 D$ = CHR$(4)
150 PRINT D$"BLOAD TRK.OBJ1"
160 HOME : VTab 5: INVERSE
170 PRINT "LEGEN SIE DIE ZU INITIALISIERENDE"
180 PRINT "DISKETTE EIN"
190 NORMAL
200 PRINT : PRINT : PRINT
210 PRINT "DRUECKEN SIE EINE TASTE ": GET Z$
220 PRINT
230 REM *** 5 NEUE TRACKS ***
240 POKE 25,1: POKE 27,0: POKE 28,0
250 POKE 29,0:T = 35
260 POKE 26,T
270 PRINT "TRACK ",T
280 CALL 847
290 IF PEEK(29) > 0 THEN GOTO 330
300 T = T + 1
310 IF T = 40 GOTO 340
320 GOTO 260
330 PRINT : PRINT : FLASH: PRINT "DISK WRITE PROTECTED": NORMAL: END
340 REM *** AENDERN DES DOS AUF DER DISKETTE ***
350 POKE 25,1: POKE 26,0: POKE 27,8: POKE 28,1
360 CALL 768
370 POKE 4350,40: POKE 28,2
380 CALL 768
390 POKE 26,2: POKE 27,2: POKE 28,1
400 CALL 768
410 POKE 4335,40: POKE 28,2
420 CALL 768
430 POKE 26,1: POKE 27,13: POKE 28,1
440 CALL 768
450 POKE 4277,160: POKE 28,2
460 CALL 768
470 REM *** SECTOR BIT MAP UPDATEN ***
480 POKE 26,17: POKE 27,0: POKE 28,1
490 CALL 768
500 RESTORE
510 FOR I = 0 TO 9
520 READ W
530 POKE W,255
540 NEXT
550 POKE 4148,40
560 POKE 28,2
570 CALL 768
580 PRINT : PRINT : PRINT
590 INVERSE : PRINT " UPDATE BEENDET "
600 PRINT "ENTNEHMEN SIE DIE DISKETTE": NORMAL
610 END
620 DATA 4292,4293,4296,4297,4300,4301,4304,4305,4308,4309
```

Bild 2. Das Basic-Programm gestattet ein benutzerfreundliches Umstellen von 35-Spur-Disketten auf 40 Spuren

Herwig Feichtinger

Programmtext-Editor

Das folgende Apple-II-Assemblerprogramm dient dazu, Texte in Maschinensprache-Software auf einfache Weise ändern zu können, z. B. um englischsprachige Software ins Deutsche zu übersetzen.

Schon in mc 1/1984 war ein kurzes Apple-Programm abgedruckt, mit dem man sich gewisse Speicherbereiche als ASCII-Zeichenfolgen ansehen konnte. Will man solche Texte ändern, so wäre es ziemlich umständlich, das Zeichen für Zeichen mit Hilfe einer ASCII-Tabelle zu machen. Einfacher geht es mit dem hier abgedruckten Programmtext-Editor, der Speicherbereiche in ASCII anzeigt und auch das Überschreiben mit neuen Zeichenfolgen gestattet.

Das Bild zeigt das mit dem Lazer-Assembler „Lisa“ erstellte Listing. Am schnellsten erfolgt die Eingabe, indem man mit CALL-151 zum Monitor geht und dann die hexadezimalen Bytes in Gruppen von je etwa drei bis vier Zeilen eingibt:

8400: A9 23 8D F9...A1 98
8420: 4C 69 FF 20 usw.

Mit BSAVE PTE,A\$8400,L\$D4 kann man das Programm auf Disk speichern, so daß man es später einfach mit BRUN PTE starten kann (das geht aber auch mit 8400 G, wenn es schon im Speicher steht).

Nach dem Start erscheint die Meldung „CTRL-Y aktiviert“, und man kann eine beliebige Hex-Startadresse eingeben, z. B. 1A00, gefolgt von CTRL-Y und Return. Nun werden die 16 Bytes ab dieser Adresse in einer Zeile als ASCII-Zeichenfolge dargestellt. Nicht darstellbare CTRL-Zeichen erscheinen als Punkte auf dem Schirm, Kleinbuchstaben werden als solche ausgegeben (sinnvollerweise verwendet man einen entsprechenden Zeichengenerator, vgl. mc 6/1983). Die

Leertaste stellt die nächsten 16 Bytes dar.

Will man nun in der letzten Zeile etwas ändern bzw. ab der zuletzt angezeigten Adresse einen neuen Text eingeben, so muß man sich zuerst entscheiden, ob der neue Text mit Bit 7 = 0 oder 1 abgespeichert werden soll. Für Bit 7 = 0 gibt man Shift-7 ein ('), sonst Shift-2 ("). Der Cursor steht nun unter dem ersten Zeichen

der letzten Zeile. Mit der Apple-Taste „Pfeil rechts“ kann man jetzt bis zu der ersten zu ändernden Position vorrücken und den Text eingeben. Korrekturen sind wie gewohnt mit den beiden Pfeiltasten möglich.

Der Eingabemodus wird beendet, sobald man das 16. Byte der Zeile erreicht hat oder vorzeitig Return drückt (Return selbst wird nicht abgespeichert). Zur Kontrolle zeigt das Programm anschließend die komplette Zeile nochmals an. Drückt man eine Taste außer der Leertaste, Shift-2 und Shift-7, so gelangt man zum Apple-Monitor zurück. Mit Shift-7 oder Shift-2 könnte man die Zeile nochmals überschreiben; mit der Leertaste rückt man 16 Bytes weiter.

Das Programm ist insbesondere geeignet, um englischsprachige Software ins Deutsche zu übersetzen und, sofern es die Tastatur und der Zeichengenerator zulassen, in Versalien geschriebene Texte in gemischte Groß- und Kleinschreibung umzusetzen. Für Maschinensprache-Programmierer ist sicher noch die Routine PRTSTR interessant, die sich auch für andere Zwecke gut anwenden läßt: Nach JSR PRTSTR folgt unmittelbar der auszugebende Text, abgeschlossen mit hex 98 als Endmarkierung.

0800	1	PNT	EPZ	\$FE		;TEXTZEIGER
0800	2	FLG	EPZ	\$FD		;BIT-7-FLAG
0800	3	ADR	EPZ	\$3C		;VOM MON.
0800	4	SAVY	EPZ	\$35		;Y-REGISTER
0800	5	YVEC	EQU	\$3F9		;CTRL-Y
0800	6	KEY	EQU	\$FD0C		;TASTE LESEN
0800	7	OUT	EQU	\$FDED		;VIDEOAUSG.
0800	8	MON	EQU	\$FF69		;MONITORSTART
0800	9	CR	EQU	\$FD8E		;CR AUSGEBEN
0800	10	PRTAX	EQU	\$F941		;HEX-ADR.AUSG.
8400	11		ORG	\$8400		
8400	12		OBJ	\$800		;WEGEN ASSEMBLER
8400	13					;CTRL-Y-VEKTOR INSTALLIEREN
8400	14	INIT	LDA	\$EDIT		;CTRL-Y-
8402	15		STA	YVEC		;VEKTOR
8405	16		LDA	/EDIT		
8407	17		STA	YVEC+1		
840A	18		JSR	PRTSTR		
840D	19		HEX	8D		
840E	20		ASC	"CTRL-Y AKTIVIERT!"		
8411						
8414						
8417						
841A						
841D						
841F	21		HEX	98		
8420	22		JMP	MON		;ZUM MONITOR
8423	23					
8423	24					
8426	25					
8428	26					
842A	27					
842D	28					
842F	29					
8432	30					

Assembler-Listing des Apple-II-Programmtext-Editors. Ohne allzu große Änderungen ist er auch für andere 6502-Computer übernehmbar

Catalog-Stopp

Wie oft schon hat ein Apple- oder ITT-2020-Besitzer sich ärgern müssen, wenn bei Benutzung von DOS 3.2 oder 3.3 nach Eingabe von CATALOG eine lange Liste von mehreren Bildschirmseiten folgt und der Name des gesuchten Files längst wieder im Jenseits verschwunden ist. Wie ärgerlich, wenn dann auf LOAD filename die bekannte Fehlermeldung FILE NOT FOUND erscheint, nur weil man bei der Eingabe vielleicht ein nicht zu erinnerndes Zeichen vergessen hat! Die Methode, bei Auftauchen des gesuchten Files den CATALOG über Reset anzuhalten und per 3D0G wieder nach DOS zu gelangen, versagt, weil offenbar einige Zeiger „verbogen“ sind.

Ein kleines Programm schafft Abhilfe. Glücklicherweise paßt es leicht in den 33-Byte-DOS-Freiraum \$BCDF...\$BCFF, so daß es durch DOS-HIMEM automatisch geschützt bleibt und durch INIT dauerhaft auf einer neuen Diskette gespeichert bleibt:

```
AE39: 20 DF BC JSR   $BCDF
BCDF: 20 0C FD JSR   $FD0C (RDKEY)
BCE2: C9 D3      CMP  #$D3
BCE4: D0 06      BNE  $BCEC
BCE6: A2 06      LDX  #$06
BCE8: 68         PLA
BCE9: CA         DEX
BCEA: D0 FC      BNE  $BCE8
BCEC: 60         RTS
```

Jeweils nach 20 (abänderbar in \$ADA4 und \$AE3D) ausgegebenen Zeilen wartet DOS auf einen Tastendruck, um den CATALOG weiter abzuarbeiten. Ein Sprung in die neue Unteroutine \$BCDF fragt nun aber zusätzlich, ob die S-Taste (= \$D3, S für Stop) gedrückt wurde. Wenn ja, so müssen nur noch die letzten drei Rücksprungadressen im Stack durch PLA vernichtet werden, und CATALOG wird gestoppt. Nun kann der Dateiname gelesen und zeichenrichtig eingegeben bzw. per ESC-Prozedur übernommen werden. Eine willkommene Hilfe für Bearbeiter von Disketten mit häufig verändertem Inhalt!

Wolfgang Jessel

Literatur

[1] Worth, D. und Lechner, P.: Beneath Apple DOS. Quality Software, 6660 Reseda Blvd., Suite 105, Reseda, CA 91 335, USA.

```
8434 20A284 31 ED1 JSR DISP ;SP.ANZEIGEN
8437 C8 32 INY
8438 C010 33 CPY £16 ;16 BYTES
843A D0F8 34 BNE ED1 ;PRO ZEILE
843C 208EFD 35 JSR CR
843F 200CFD 36 JSR KEY ;TASTE
8442 C9A0 37 CMP £$A0 ;SPACE?
8444 D00D 38 BNE ED2
8446 A53C 39 LDA ADR ;JA
8448 690F 40 ADC £15 ;+16 (C=1)
844A 853C 41 STA ADR
844C 9002 42 BCC ED3
844E E63D 43 INC ADR+1
8450 4C2684 44 ED3 JMP ED0
8453 C9A2 45 ED2 CMP £$A2 ;"?
8455 D004 46 BNE ED4
8457 A9FF 47 LDA £$FF ;BIT7=1
8459 D009 48 BNE ED5
845B C9A7 49 ED4 CMP '' ;'?
845D F003 50 BEQ ED4A
845F 4C69FF 51 JMP MON
8462 A97F 52 ED4A LDA £$7F ;BIT7=0
8464 85FD 53 ED5 STA FLG ;FLAG
8466 A005 54 LDY £5 ;5 SPACES
8468 A9A0 55 SPC LDA £$A0
846A 20EDFD 56 JSR OUT
846D 88 57 DEY
846E D0F8 58 BNE SPC
8470 20CC84 59 ED6 JSR GET ;TASTE
8473 C98D 60 CMP £$8D ;RETURN?
8475 F0AC 61 BEQ EDIT
8477 C988 62 CMP £$88 ;BACKSPACE?
8479 F012 63 BEQ BS
847B C995 64 CMP £$95 ;CUR.RECHTS?
847D F019 65 BEQ RGT
847F 20EDFD 66 JSR OUT ;ANZEIGEN
8482 25FD 67 AND FLG ;UND
8484 913C 68 STA (ADR),Y ;ABSPEICHERN
8486 C8 69 INY
8487 C010 70 CPY £16 ;ZEILENENDE?
8489 F098 71 BEQ EDIT ;JA
848B D0E3 72 BNE ED6 ;NEIN
848D C000 73 BS CPY £0 ;ZEILENANF.?
848F F0DF 74 BEQ ED6 ;JA
8491 88 75 DEY ;NEIN
8492 20EDFD 76 JSR OUT
8495 4C7084 77 JMP ED6
8498 C010 78 RGT CPY £16 ;ZEILENENDE?
849A F087 79 BEQ EDIT ;JA
849C 20A284 80 JSR DISP ;Z.ANZEIGEN
849F C8 81 INY
84A0 D0CE 82 BNE ED6
84A2 83 ;ZEICHEN AUS SPEICHER ANZEIGEN
84A2 B13C 84 DISP LDA (ADR),Y
84A4 0980 85 ORA £$80 ;BIT7=1
84A6 C9A0 86 CMP £$A0
84A8 B002 87 BCS DISP1
84AA A9AE 88 LDA ''
84AC 4CEDFD 89 DISP1 JMP OUT ;RTS
84AF 90 ;TEXTAUSGABE
84AF 91 ;MIT JSR PRTSTR, DANN TEXT BIS $98
84AF 68 92 PRTSTR PLA ;ADR.DES
84B0 85FE 93 STA PNT ;STRINGS
84B2 68 94 PLA ;HOLEN
84B3 85FF 95 STA PNT+1
84B5 E6FE 96 PRT1 INC PNT ;ZEIGER
84B7 D002 97 BNE PRT2 ;INKREM.
84B9 E6FF 98 INC PNT+1
84BB A000 99 PRT2 LDY £0
84BD B1FE 100 LDA (PNT),Y ;STRING
84BF C998 101 CMP £$98 ;BIS 'TYA'
84C1 F006 102 BEQ PRT3 ;DRUCKEN
84C3 20EDFD 103 JSR OUT
84C6 4CB584 104 JMP PRT1
84C9 6CFE00 105 PRT3 JMP (PNT)
84CC 106 ;TASTE LESEN, Y RETTEN
84CC 8435 107 GET STY SAVY
84CE 200CFD 108 JSR KEY
84D1 A435 109 LDY SAVY
84D3 60 110 RTS
111 END
```


Herwig Feichtinger

Prüfsummen- Programm für den Apple-II

Beim Abtippen von Maschinenprogrammen aus Zeitschriften können Fehler üble Folgen haben. Deshalb ergänzt mc solche Hexdumps oft mit Prüfsummen. Ein Programm, das diese 16-Bit-Summen zeilenweise errechnet, bringen wir hier für den Apple-II.

Das Programm in Bild 1 wurde mit dem Lazer-Assembler „Lisa“ assembliert. Es ist im Source- und Objektformat auch auf unserer Apple-Sammeldiskette 6 enthalten (Best.-Nr. AP 006 beim Franzis-Software-Service, 20 DM) und für

den Apple-II mit min. 48 KByte RAM und DOS 3.3 ausgelegt. Die Startadresse und Länge des auszugebenden Hexdumps holt sich das Programm aus den DOS-Speicherzellen AA72/AA73 und AA60/AA61. Dort ste-

```
0800- 20 9E 08 8D 8D C8 C5 D8 +=0445
0808- C4 D5 CD D0 8D 8D 84 D0 +=05A4
0810- D2 A3 B2 8D 8D 98 AD 72 +=04F8
0818- AA 85 3C AD 73 AA 85 3D +=03F7
0820- A5 3D A6 3C 20 41 F9 A9 +=03C7
0828- 00 85 3E 85 3F A9 AD 20 +=02FD
0830- ED FD A2 08 A9 A0 20 ED +=04EA
0838- FD A0 00 B1 3C 20 DA FD +=0481
0840- B1 3C 18 65 3E 85 3E 90 +=02FB
0848- 02 E6 3F E6 3C D0 02 E6 +=0401
0850- 3D AD 60 AA 18 6D 72 AA +=0395
0858- 85 40 AD 73 AA 6D 61 AA +=0407
0860- C5 3D D0 06 A5 40 C5 3C +=03BE
0868- F0 09 CA D0 C7 20 8B 08 +=040D
0870- 4C 20 08 20 8B 08 20 9E +=01E5
0878- 08 84 D0 D2 A3 B0 8D 8D +=049B
0880- C6 C5 D2 D4 C9 C7 A1 8D +=05EF
0888- 8D 98 60 20 9E 08 A0 AB +=0396
0890- BD 98 A5 3F A6 3E 20 41 +=037E
0898- F9 A9 8D 4C ED FD 68 85 +=0552
08A0- 40 68 85 41 E6 40 D0 02 +=0366
08A8- E6 41 A0 00 B1 40 C9 98 +=0419
08B0- F0 06 20 ED FD 18 90 EC +=0494
08B8- 6C 40 00 +=00AC
```

Bild 2. Hier hat sich das Programm aus Bild 1 selbst ausgedruckt

hen diese Daten automatisch, wenn man den auszudruckenden Speicherbereich im BLOAD lädt.

Die Bedienung erfolgt also normalerweise so:

```
0800      1  ;HEXDUMP-AUSGABE MIT PRUEFSUMMEN
0800      2  ;H.FEICHTINGER/6.4.1984
0800      3  OUT  EQU $FDED      ;ZEICHEN AUSGEBEN
0800      4  PRBYTE EQU $FDDA     ;BYTE AUSGEBEN
0800      5  PRTAX EQU $F941     ;A,X HEX AUSGEBEN
0800      6  START EQU $AA72     ;STARTADRESSE
0800      7  ENDE EQU $AA60      ;FILE-LAENGE
0800      8  PNT  EPZ $3C        ;ADR.-ZEIGER
0800      9  SUM  EPZ $3E        ;PRUEFSUMME
0800     10  TADR EPZ $40        ;TEXTZEIGER
0800     11      ORG $800
0800     12  JSR PRTSTR
0800     13  HEX 8D8D
0800     14  ASC "HEXDUMP"
0800     15  HEX 8D8D84          ;CTRL-D F.DOS
0800     16  ASC "PR#2"          ;DRUCKER-SLOT 2
0800     17  HEX 8D8D98
0800     18  LDA START           ;STARTADRESSE
0800     19  STA PNT            ;SETZEN
0800     20  LDA START+1
0800     21  STA PNT+1
0800     22  LINE LDA PNT+1
0800     23  LDX PNT
0800     24  JSR PRTAX           ;ADRESSE
0800     25  LDA #0             ;PRUEFSUMME
0800     26  STA SUM            ;LOESCHEN
0800     27  STA SUM+1
0800     28  LDA "-"
0800     29  JSR OUT
0800     30  LDX #8             ;8 BYTE/ZEILE
0800     31  BYTE LDA #$A0       ;SPACE
0800     32  JSR OUT
0800     33  LDY #0
0800     34  LDA (PNT),Y         ;BYTE
0800     35  JSR PRBYTE         ;DRUCKEN
0800     36  LDA (PNT),Y
0800     37  CLC
0800     38  ADC SUM            ;PRUEFSUMME
0800     39  STA SUM            ;ERRECHNEN
0800     40  BCC NADR           ;KEIN CARRY
0800     41  INC SUM+1
0800     42  NADR INC PNT        ;ADRESSE
0800     43  BNE LADR           ;INKREMENTIEREN
0800     44  INC PNT+1
0800     45  LADR LDA ENDE      ;ENDADRESSE
0854     18      46      CLC
0855     47      46      ADC START      ;ERRECHNEN
0858     48      47      STA TADR
085A     49      48      LDA START+1
085D     50      49      ADC ENDE+1
0860     51      50      CMP PNT+1
0862     52      51      BNE NOC
0864     53      52      LDA TADR
0866     54      53      CMP PNT
0868     55      54      BEQ READY
086A     56      55      NOC
086B     57      56      DEX
086D     58      57      BNE BYTE
0870     59      58      JSR PRSUM
0873     60      59      JMP LINE
0876     61      60      JSR PRSUM
0879     62      61      JSR PRTSTR
087A     63      62      HEX 84
087D     64      63      ASC "PR#0"
087E     65      64      HEX 8D8D
0880     66      65      ASC "FERTIG!"
0883     67      66      HEX 8D8D98
0886     68      67      RTS
0888     69      68      PRSUM JSR PRTSTR
088B     70      69      ASC " += "
0891     71      70      HEX 98
0892     72      71      LDA SUM+1
0894     73      72      LDX SUM
0896     74      73      JSR PRTAX
0899     75      74      LDA #$8D
089B     76      75      JMP OUT
089F     77      76      PRTSTR PLA
08A1     78      77      STA TADR
08A2     79      78      PLA
08A4     80      79      STA TADR+1
08A6     81      80      PRT1 INC TADR
08A8     82      81      BNE PRT2
08AA     83      82      INC TADR+1
08AC     84      83      LDY #0
08AE     85      84      LDA (TADR),Y
08B0     86      85      CMP #$98
08B2     87      86      BEQ PRT3
08B5     88      87      JSR OUT
08B6     89      88      CLC
08B8     90      89      BCC PRT1
08B8     91      90      PRT3 JMP (TADR)
08B8     91      91      ENDE
```

Bild 1. Assemblerlisting des Hex-Dump-Programms mit Prüfsummen-Ausgabe

1. Mit CALL-151 in den Monitor gehen.
2. Programm in Bild 1 z. B. von Diskette laden.
3. Auszugebendes Programm von Diskette laden.
4. Dump-Programm mit 800G starten.

Natürlich ist es ebenso gut möglich, Startadresse und Länge von Hand in die genannten Speicherzellen einzugeben. Es ist einzusehen, daß sich Druckprogramm und auszugebendes Programm nicht überschneiden dürfen. Sonst muß

man das Druckprogramm ab einer anderen Adresse neu assemblieren.

Bild 2 zeigt das Ausgabeformat; hier hat sich das Dump-Programm selbst ausgedruckt. Selbstverständlich ist es auch möglich, 16 statt 8 Bytes pro Zeile zu drucken; dann muß man das Byte 08 an Adresse 0833 auf hex 10 ändern. Ebenso einfach ist die Änderung des Drucker-Slots (hier 2): Der Slot-Nummer wird ein B vorangestellt, das resultierende Hex-Byte (B1 für Slot 1) steht an der Adresse 0812. Will man dagegen die Ausgabe nur auf den Bildschirm leiten, so ist hex B0 bei 0812 einzugeben.

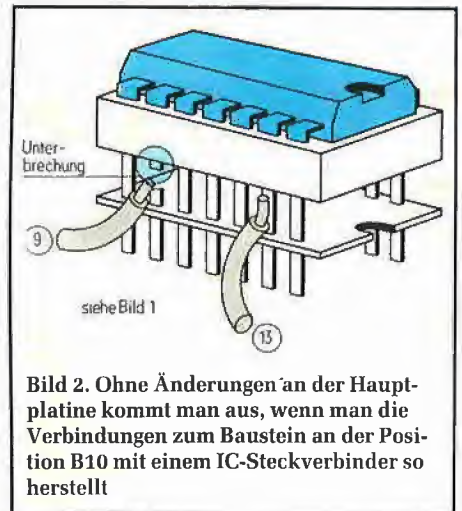


Bild 2. Ohne Änderungen an der Hauptplatine kommt man aus, wenn man die Verbindungen zum Baustein an der Position B10 mit einem IC-Steckverbinder so herstellt

Tastaturpuffer für Apple-II

Ein Nachteil des Apple-II ist der fehlende Tastaturpuffer. Läuft die Floppy (mit DOS) oder ein langsames Programm, so wird die Tastatur nicht abgefragt, und einzelne Zeichen können verlorengehen. Die in Bild 1 dargestellte Schaltung behebt diesen Mangel. Sie speichert bis zu 16 Zeichen. Völlig im Hintergrund arbeitend, erlaubt sie eine sehr viel flüssigere Dateneingabe vor allem bei DOS-Befehlen.

Hauptbestandteil der Schaltung ist der Baustein CD40105, ein FIFO-Stack mit einer Datenbreite von 4 Bit und einer Stacktiefe von 16. Er ist für die parallele asynchrone Datenübertragung gedacht und durch seinen Aufbau besonders gut für diese Anwendung geeignet.

Angeschlossen wird die Schaltung an dem kurzen 16poligen Kabel, das Tastatur und Rechner miteinander verbindet. Bis auf die Datenleitungen bleiben alle Verbindungen erhalten. Der Anschluß wird am besten mit einem Flachbandkabel durchgeführt. Jeder über die Leitung „Strobe“ (Pin 2) ankommende Impuls veranlaßt den Stack, die Daten von der Tastatur zu übernehmen und den Stackpointer um eins hochzuzählen.

Der „DOR“-Ausgang der ICs geht dann auf High-Pegel und zeigt damit an, daß der Stack nicht leer ist. Dieses Signal, um zwei Gatterlaufzeiten (CD 4093) verzögert, ersetzt das Strobe-Signal im Rechner. Es wird durch das höchste Bit

am Datenbus dargestellt und durch eines der Flipflops von IC 74LS74, an der Stelle B10 der Hauptplatine, gesteuert. Dieser Anschluß (Pin 9 des ICs) ist nun nicht mehr erforderlich und muß getrennt werden. Dazu ist der in Bild 2 dargestellte Aufbau empfehlenswert, damit keine Änderung an der Hauptplatine des Rechners erforderlich ist.

Vom Anschluß 13 kann gleich das zweite erforderliche Steuersignal, das „Clear Keyboard Strobe“, abgegriffen werden. Dieses Signal wird in Programmen durch Ansprechen der Adressen \$C010 gesetzt, wenn ein Tastendruck abgearbeitet worden ist. In dieser Schaltung wird es dazu benutzt, den Stack wieder um eins herunterzuzählen.

In diesem Punkt liegt ein kleines Problem, denn nicht bei allen Programmen kommt für jedes Zeichen genau ein Impuls auf dieser Leitung. Während bei Basic, Pascal und nahezu allen mir bekannten Dienstprogrammen keine Schwierigkeiten auftreten, benutzen einige Spiele wie „Head On“, „Panic“ usw. die Leitung überhaupt nicht. Der „EDASM“ hingegen benutzt sie bei der Befehlsabarbeitung häufig mehrmals.

Aus diesem Grund kann der Puffer außer Betrieb gesetzt werden. Öffnet man den Schalter „Buffer on/off“, dann wird bei jeder ansteigenden Flanke der Stack zurückgesetzt. Somit bleibt nur der letzte Tastendruck gespeichert, und der Original-Zustand des Apple ist simuliert. Die preiswerte Schaltung (Bauteilewert ca. 20 DM) ist sicherlich auch für andere Rechnerarten geeignet. Sie läßt einen Bedienungskomfort zu, wie man ihn sonst nur bei kommerziellen Tastaturen gewohnt ist.

Heinz-Jörg Schröder

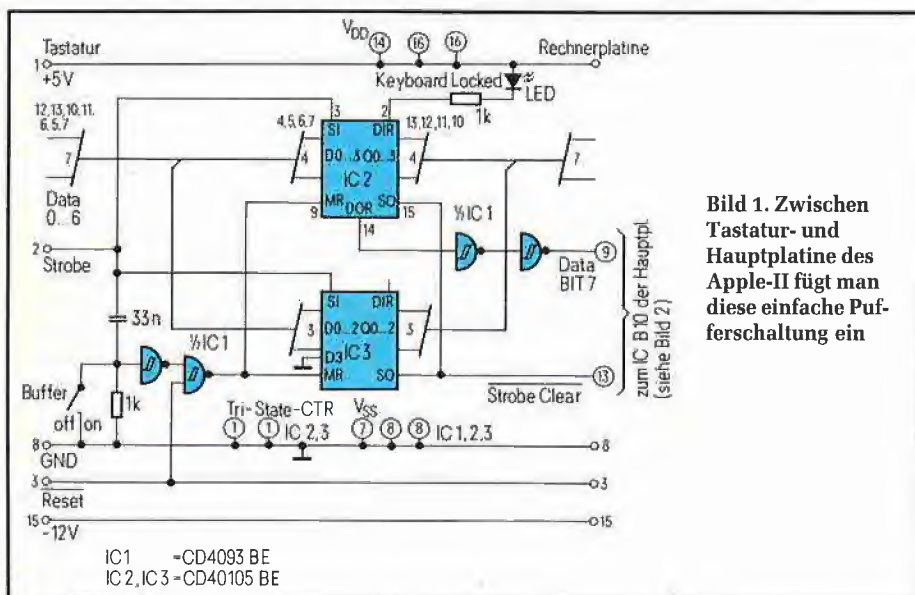


Bild 1. Zwischen Tastatur- und Hauptplatine des Apple-II fügt man diese einfache Puffererschaltung ein

Dirk Fischer

Relocator für den Apple-II

Die meisten Programme in der Maschinensprache des Prozessors 6502 laufen nur in dem Speicherbereich, für den sie geschrieben sind. Sollen sie in einen anderen Bereich verschoben werden, muß man absolute Adressen umrechnen. Diese Arbeit kann der Computer selbst übernehmen – mit einem Relocator.

Der in Bild 1 abgedruckte Relocator

- verschiebt ein Programm,
- berechnet absolute Adressen für den neuen Speicherbereich,
- erstellt eine Liste mit den umgerechneten Adressen,
- erkennt ungültige Operationscodes (Fehlermeldung).

Ungültige Operationscodes treten z. B. in Datenbereichen (Tabellen mit ASCII-Zeichen usw.) auf. Derartige Abschnitte lassen sich überspringen, da hier ja keine Umrechnung erforderlich ist.

Bild 2 zeigt ein Ausführungsbeispiel. Zunächst gibt man Anfangs- und Endadresse des gesamten zu verschiebenden Bereichs (einschließlich Daten) ein. Es folgen der Beginn des neuen Speicherbereichs und die Anfangs- bzw. Endadresse des Programmbereichs, in dem Adressen umzurechnen sind. Als Ergebnis erhält man die Liste der geänderten Bytes. In diesem Fall hat sich der Relocator selbst verschoben. Seine ursprüngliche Startadresse ist hex 800. Nach dem Verschieben kann er mit 1000 G vom Monitor aus gestartet werden. Die Anpassung an andere Systeme ist wenig sinnvoll, da intensiver Gebrauch von Unterprogrammen des Apple-Monitors gemacht wird.

```
START ORIGINAL.....800
ENDE ORIGINAL.....81D
START NEU.....1000
START UMRECHNUNG.....800
ENDE UMRECHNUNG.....987
```

PROGRAMM VERSCHOBEN IN BEREICH
1000 BIS 131D

FOLGENDE BYTES HABEN SICH
GEÄNDERT:

```
1012-11 (09)
101D-11 (09)
102B-11 (09)
1031-11 (09)
103A-11 (09)
1066-11 (09)
1075-11 (09)
1091-12 (0A)
10A7-11 (09)
1102-11 (09)
1114-11 (09)
112C-12 (0A)
116E-10 (08)
```

Bild 2. Beispiel eines Programmdurchlaufs: Hier hat sich das Programm selbst verschoben

Nicht alle Assemblerprogramme funktionieren noch nach dem Verschieben mit dem Relocator. In solchen Fällen muß man das Programm mit Hilfe eines Disassemblers analysieren. Insbesondere müssen Unterprogrammsprünge innerhalb des verschobenen Programms von Hand geändert werden, da sie nicht automatisch korrigiert werden. Darauf wurde deswegen verzichtet, weil Unterprogrammsprünge häufig auf feste Adressen gerichtet sind (z. B. im Monitor).

0800-- 20 58 FC A9 05 85 20 A9 23 85 21 20 BE FD A2 00	0990-- AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0810-- BD 88 09 20 ED FD E8 D0 F7 A2 D0 BD 88 09 20 ED	09A0-- AA AA AA AA AA AA BD AA A0 A0 A0 A0 A0 A0 A0 A0
0820-- FD E8 D0 F7 A9 0E 20 70 09 85 F8 B6 F9 A9 0F 20	09B0-- A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
0830-- 70 09 85 FA B6 FB A9 10 20 70 09 85 FC B6 FD A2	09C0-- A0 A0 AA BD AA A0 CB A0 C5 A0 D8 A0 AD A0 D2 A0
0840-- 03 B5 F8 95 3C CA 10 F9 A5 FC 85 42 A5 FD 85 43	09D0-- C5 A0 CC A0 CF A0 C3 A0 C1 A0 D4 A0 CF A0 D2 A0
0850-- A0 00 20 2C FE A5 FC 3B E5 F8 85 F6 A5 FD E5 F9	09E0-- AA BD AA A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0
0860-- 85 F7 A9 11 20 70 09 18 65 F6 85 3A 8A 65 F7 85	09F0-- A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 A0 AA BD
0870-- 3B A9 12 20 70 09 18 65 F6 85 FE 8A 65 F7 85 FF	0A00-- AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0880-- A2 00 A1 3A 85 F2 20 BE FB C9 10 D0 1C A2 E8 BD	0A10-- AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
0890-- 1D 0A 20 ED FD E8 D0 F7 3B A5 3A E5 F6 AA A5 3B	0A20-- A0 A0 A0 A0 A0 A0 A0 AB C3 A9 A0 C4 C9 D2 CB A0 C6
08A0-- E5 F7 20 41 F9 4C 60 09 00 A5 2F C9 02 D0 36 A0	0A30-- C9 D3 C3 CB C5 D2 BD BD A0 A0 A0 A0 A0 A0 A0 A0
08B0-- 02 B1 3A 85 F3 B8 B1 3A 85 F2 3B A5 F9 C5 F3 90	0A40-- A0 A0 A0 B0 B4 AE B0 B9 AE B8 B1 BD BD BD BD BD
08C0-- 08 D0 22 A5 F2 C5 F8 90 1C A5 FB C5 F3 90 16 D0	0A50-- D3 D4 C1 D2 D4 A0 CF D2 C9 C7 C9 CE C1 CC AE AE
08D0-- 06 A5 FA C5 F2 90 0E A5 F2 18 65 F6 91 3A CB A5	0A60-- AE AE AE AE BD C5 CE C4 C5 A0 A0 CF D2 C9 C7 C9
08E0-- F3 65 F7 91 3A 20 53 F9 85 3A 84 3B 98 C5 FF 90	0A70-- CE C1 CC AE AE AE AE AE AE BD D3 D4 C1 D2 D4 A0
08F0-- 8F D0 08 A5 3A C5 FE 90 87 F0 85 20 42 FC A2 DE	0A80-- CE C5 D5 AE AE AE AE AE AE AE AE AE AE AE BD D3
0900-- BD DA 09 20 ED FD E8 D0 F7 A6 FC A5 FD 20 41 F9	0A90-- D4 C1 D2 D4 A0 D5 CD D2 C5 C3 CB CE D5 CE C7 AE
0910-- A2 FB BD DF 09 20 ED FD E8 D0 F7 18 A5 FA 65 F6	0AA0-- AE AE AE BD C5 CE C4 C5 A0 A0 D5 CD D2 C5 C3 CB
0920-- AA A5 FB 65 F7 20 41 F9 A2 D9 BD 06 A0 20 ED FD	0AB0-- CE D5 CE C7 AE AE AE AE BD BD D0 D2 CF C7 D2 C1
0930-- E8 D0 F7 A2 EE 20 A8 FC E8 D0 FA A2 03 B5 FC 95	0AC0-- CD CD A0 D6 C5 D2 D3 C3 CB CF C2 C5 CE A0 C9 CE
0940-- 3C CA 10 F9 A5 F8 85 42 A5 F9 85 43 A0 00 20 36	0AD0-- A0 C2 C5 D2 C5 C9 C3 CB BD BD A0 C2 C9 D3 A0 BD
0950-- FE A9 00 85 20 A9 28 85 21 20 BE FD 4C 69 FF 00	0AE0-- BD C6 CF CC C7 C5 CE C4 C5 A0 C2 D9 D4 C5 D3 A0
0960-- A5 3B C5 FF 30 06 A5 3A C5 FE 10 8F 4C 62 08 00	0AF0-- CB C1 C2 C5 CE A0 D3 C9 C3 CB BD C7 C5 C1 C5 CE
0970-- 85 25 20 BE FD A9 14 85 24 20 6F FD 20 C7 FF 20	0B00-- C4 C5 D2 D4 BA BD BD 55 4E 47 55 45 4C 54 49 47
0980-- A7 FF A5 3E A6 3F 60 00 AA AA AA AA AA AA AA AA	0B10-- 45 52 60 4F 50 43 4F 44 45 A0 C9 CE A0 60

Bild 1. Hilfestellung beim Verschieben von Maschinenprogrammen leistet dieser Relocator für den Apple-II

Bernhard Kapitza

Für Apple-II und MX-82F/T:

Spooler beschleunigt Druckausgabe

In mc 6/82 stellten wir ein Spooler-Programm für den TRS-80 vor. Ein solches Programm ermöglicht eine schnellere Druckausgabe, da die auszugebenden Zeichen in einen Puffer geschrieben werden, von wo sie eine Interruptroutine quasi „zwischendurch“ abholt. Hier ist nun die Übersetzung des damals veröffentlichten Spoolers für den Apple-II und den Drucker MX-80/FT.

Als Interruptquelle dient eine I/O-Karte mit dem VIA 6522. Dieser Baustein besitzt unter anderem einen programmierbaren 16-Bit-Timer. Das 16-Bit-Register kann mit einem beliebigen Wert vorbesetzt werden. Im „Free-running-mode“ wird nun dieser Wert mit jedem Systemtakt um 1 verringert. Wenn der Wert 0 erreicht ist wird im Interrupt-Flag-Register (IFR) ein Bit auf 1 gesetzt. Bei entsprechender Programmierung des Interrupt-Enable-Registers (IER) wird ein Interrupt erzeugt, und der Prozessor beginnt mit der Verarbeitung des Interruptprogramms. Der Vorteil des „Free-running-modes“ ist, daß bei Erreichen der 0 das 16-Bit-Register automatisch wieder mit dem ursprünglichen Wert geladen und wieder bis 0 dekrementiert wird. Diese Betriebsart wird über das ACR-Register gesetzt [2].

Für das Spoolprogramm kann auch jede andere Interruptquelle benutzt werden; es müssen jedoch unter Umständen einige Programmteile, hauptsächlich SPRINT, verändert werden. Das Programm (Bild) wird vom Basic aus durch drei Kommandos gesteuert: &POS: Print On Spooler Die Ausgabe erfolgt nicht mehr über den Bildschirm, sondern die Zeichen werden in den Spoolpuffer geschrieben und durch das Interruptprogramm über den Drucker ausgegeben.

&NORMAL:
&END:

Die Ausgabe der Zeichen erfolgt wieder über den Bildschirm. Der Unterschied zwischen den beiden letzten Kommandos ist, daß bei &NORMAL weiterhin die noch im Puffer stehenden Zeichen ausgedruckt werden; &END beendet den Druckvorgang.

Hinweise zum Listing

Zum besseren Verständnis folgen Erläuterungen zu einigen benutzten Namen im Programm:

AUS: Anzahl der zu druckenden Zeichen im Puffer
FREI: Anzahl noch freier Plätze im Puffer
BUFFER: Startadresse des Puffers
BUFFEND: Endadresse des Puffers
LZEIG: Adresse des nächsten zu lesenden Zeichens
SZEIG: Adresse der nächsten freien Stelle
DTEST: Speicherstelle, die anzeigt, ob der Drucker bereit ist, neue Daten aufzunehmen. Die im Programm benutzte Adresse ist unter Umständen nur für das Original-Epson-Interface gültig; das Bedienungshandbuch gibt darüber Auskunft. Gleiches gilt für:
DRUCK: Alle hier abgespeicherten Zeichen werden an den Drucker gesendet

BEGIN: Das „&“-Kommando im Applesoft-Basic bewirkt einen unbedingten Sprung zur Stelle \$3F5. Normalerweise steht dort ein Sprung zur Stelle \$FF58. „BEGIN“ speichert nun den Inhalt von \$3F6 (BJP) und CONTINUE um, so daß beim „&“-Kommando ein Sprung zu STRT durchgeführt wird. Durch Setzen der HIMEM-Pointer wird das Spoolprogramm vor Überschreiben geschützt [3].

STRT: Diese Routine überprüft, ob eines der drei gültigen Kommandos POS, NORMAL, END vorliegt. Da diese Kommandos Basic-Befehle sind, werden sie auch als solche, nämlich als sogenannte Tokens, im Programm abgespeichert, d. h. pro Befehlswort ein Byte [3]. Lag keiner der drei Befehle vor, so wird in die alte „&“-Kommandoroutine verzweigt (deshalb auch die Umspeicherung).

SPRINT: Speichert die Adresse der neuen Ausgaberroutine in DOSEXIT ab, der alte Wert wird in CSWSP zwischengespeichert. DOSEXIT beinhaltet die Anfangsadresse einer zur Zeit gültigen Output-Routine (vergleichbar mit der CSWL-Speicherstelle des Apple-Monitors) [4, 5]. Der Interruptzeiger \$3FE wird auf die Anfangsadresse des Interruptprogramms SINTRP gesetzt.

SPEICH: Programm, das die Daten in den Puffer schreibt.

SINTRPT: Interruptprogramm. Es wird getestet, ob ein Timer-1-Interrupt vorliegt. Ansonsten erfolgt ein Sprung zur alten Interruptroutine.

Pufferbereich kann geändert werden

Die Eingabe des Programms nimmt man am besten mit einem Assembler vor. Dabei kann dann auch die Größe des Puffers nach eigenen Erfordernissen festgelegt werden. Zu beachten ist jedoch, daß bei einer Vergrößerung des Puffers eventuell das Betriebssystem überschrieben wird (bei 48 KByte Hauptspeicher und MAXFILES 3 beginnen die DOS-File-

puffer bei \$9600). Es ist dann gegebenenfalls die Anfangsadresse des Programms zu ändern.

Probleme kann es geben, wenn Interfacekarten benutzt werden, die eigene (E)PROMs im Bereich \$C800...CFFF haben (z. B. 80-Zeichen-Karte, seriell Interface usw.). Da dieser Bereich für jeden Slot zur Verfügung steht (d. h. mehrfach belegt ist), werden diese (E)PROMs nach Bedarf aktiviert/desaktiviert. Die Interruptroutine „weiß“ bei Beendigung dann nicht mehr, welcher Speicherbaustein vor dem Interrupt gerade aktiv war [5].

Literatur

- [1] Breymann, U.: Druckausgabe nebenbei. mc 1982, Heft 6, Seite 64...68.
- [2] Rockwell-Datenblatt VIA 6522.
- [3] Basic Programming Reference Manual. Apple Computer Inc.
- [4] Beneath Apple DOS. Quality Software.
- [5] Apple II Reference Manual. Apple Computer Inc.

```

0010 .BA $B000
0020 .DS
0030
0040 ;
0050 ;
0060 ;
0070 ; APPLE+ 48k DOS 3.3 I/O Karte
0080 ; B.Kapitza Sept 82
0090
0100 LF
0110 CR
0120 HIMEM
0130 DOSEXIT
0140 BJP
0150 IRQ
0160 COUT
0170
0180 DTEST
0190 DRUCK
0200
0210 TILL
0220 TICH
0230 ACR
0240 IFR
0250 IER
0260
0270 TIME
0280 ;
0290 ;
0300 ; Initialisierung
0310 ;
0320 LDA BJP ;&-Kommando um-
0330 STA CONTINUE+1
0340 LDA BJP+1 ; speichern
0350 STA CONTINUE+2
0360 LDA #L,STRT ; und Programm vor
0370 STA $HIMEM
0380 STA BJP ; überschreiben schützen
0390
8000- AD F6 03
8003- BD 52 80
8006- AD F7 03
8009- BD 53 80
800C- A9 40
800E- B5 73
8010- BD F6 03

```

Eine schnellere Druckausgabe erreicht man mit diesem Spoolprogramm. Übrigens ist „Spool“ die Abkürzung für „Simultaneous Peripheral Output On Line“

```

0390 LDA #H,STRT
0400 STA $HIMEM+1
0410 STA BJP+1
0420 LDY #24 ;Message ausgeben
0430 LDA TBL-1,Y
0440 ORA #$80
0450 JSR COUT
0460 DEY
0470 BNE LOOP
0480 RTS
0490 TBL
0500 .BY 'treillatsni RELOOPS >>>>'
0510 ;
0520 ;&-Kommando und Test
0530 ;
0540 STRT
0550 PHA ;nächstes Zeichen
0560 PLA
0570 CMP #217 ;Token POS 'Print on SPOOLER'
0580 BEQ SPRINT
0590 CMP #157 ;NORMAL 'Ende Print in BUFFER'
0600 BEQ NPRINT
0610 CMP #128 ;END 'Ende SPOOLER u. INTERRUPT'
0620 BEQ ENDE
0630 JMP $FF5B ;wird u.U. verändert
0640 CONTINUE
0650 ;
0660 ;&END - Kommando
0670 ;
0680 ENDE
0690 SEI ;Interruptquelle
0700 LDA #201000000
0710 STA IER ;abschalten
0720 LDA ACR
0730 AND #210111111
0740 STA ACR
0750 LDA EXIT+1 ;INT-Adresse rückspeichern
0760 STA IRQ
0770 LDA EXIT+2
0780 STA IRG+1
0800 ;
0810 ;&NORMAL - Kommando
0820 ;
0830 NPRINT
0840 LDA CSMSP ;Output unspeichern
0850 STA DOSEXIT
0860 LDA CSMSP+1
0870 STA DOSEXIT+1
0880 RTS
0890 ;
0900 ;&POS - Kommando
0910 ;
0920 SPRINT
0930 LDA DOSEXIT
0940 STA CSMSP
0950 LDA DOSEXIT+1
0960 STA CSMSP+1

```

8087- A9 EB	0960	LDA #L, SPEICH	; routine mit alter Output-	86F7- 78	1590 NEXT1	SEI	;ein
8089- 8D 53 AA	0970	STA DOEXIT		86F8- CE E7 80	1600	DEC FREI	;freier
808C- A9 86	0980	LDA #H, SPEICH	; routine tauschen und speichern	86FB- AD E7 80	1610	LDA FREI	;Platz
808E- 8D 54 AA	0990	STA DOEXIT+1		86FE- C9 FF	1620	CMP #FF	;weniger
	1000			8700- D0 03	1630	BNE L1	
8091- A9 EB	1010	LDA #L, BUFFER	; BUF-Adr. in Zeiger speichern	8702- CE E8 80	1640	DEC FREI+1	
8093- 8D 10 87	1020	STA SIZEIG+1			1650 L1		
8096- 8D 64 87	1030	STA LZEIG+1		8705- EE E9 80	1660	INC AUS	;ein
8099- A9 80	1040	LDA #H, BUFFER		8708- D0 03	1670	BNE L2	;auszugebendes
809B- 8D 11 87	1050	STA SIZEIG+2		870A- EE EA 80	1680	INC AUS+1	;Zeichen mehr
809E- 8D 65 87	1060	STA LZEIG+2			1690 L2		
	1070			870D- 68	1700	PLA	;Zeichen holen
80A1- A9 00	1080	LDA #0	; Zähler AUS u. FREI vorbesetzen	870E- 48	1710	PHA	
80A3- 8D E9 80	1090	STA AUS			1720 SIZEIG		
80A6- 8D EA 80	1100	STA AUS+1		870F- 8D 00 00	1730	STA \$0	;Zeichen abspeichern
80A9- A9 FF	1110	LDA #L, BUFFEND-BUFFER		8712- AD 10 87	1740	LDA SIZEIG+1	
80AB- 8D E7 80	1120	STA FREI		8715- C9 EA	1750	CMP #L, BUFFEND	
80AE- A9 05	1130	LDA #H, BUFFEND-BUFFER		8717- AD 11 87	1760	LDA SIZEIG+2	;Bufferende
80B0- 8D E8 80	1140	STA FREI+1		871A- E9 86	1770	SBC #H, BUFFEND	
	1150			871C- EE 10 87	1780	INC SIZEIG+1	;erreicht?
80B3- AD FE 03	1160	LDA IRQ	; Interruptadresse unspeichern	871F- D0 03	1790	BNE L3	
80B6- 8D A2 87	1170	STA EXIT+1		8721- EE 11 87	1800	INC SIZEIG+2	
80B9- AD FF 03	1180	LDA IRQ+1			1810 L3		
80BC- 8D A3 87	1190	STA EXIT+2		8724- 90 0A	1820	BCC L4	;C=1 wenn BUFFEND erreicht
80BF- A9 34	1200	LDA #L, SINTRPT		8726- A9 EB	1830	LDA #L, BUFFER	
80C1- 8D FE 03	1210	STA IRQ		8728- 8D 10 87	1840	STA SIZEIG+1	;SIZEIG auf
80C4- A9 87	1220	LDA #H, SINTRPT		872B- A9 80	1850	LDA #H, BUFFER	
80C6- 8D FF 03	1230	STA IRQ+1		872D- 8D 11 87	1860	STA SIZEIG+2	;Anfang setzen
	1240				1870 L4		
80C9- A9 FF	1250	LDA #L, TIME	; Interruptzeit einspeichern	8730- 68	1880	PLA	
80CB- 8D D4 C0	1260	STA TILL		8731- 28	1890	PLP	
80CE- A9 0F	1270	LDA #H, TIME		8732- 58	1900	CLI	
80D0- 8D D5 C0	1280	STA TICH		8733- 60	1910 END1	RTS	
	1290				1930 ;		
80D3- AD DB C0	1300	LDA ACR	;Free-Running-Mode setzen		1940 ;Druck-Programm		
80D6- 09 40	1310	ORA #201000000			1950 ;		
80D8- 8D DB C0	1320	STA ACR		8734- AD DD C0	1960 SINTRPT	LDA IFR	;T1 - Interrupt?
	1330			8737- 29 40	1970	AND #201000000	
80DB- AD DE C0	1340	LDA IER	;TIMER 1 - Interrupt setzen	8739- F0 66	1980	BEQ EXIT	
80DE- 09 C0	1350	ORA #211000000		873B- 8A	1990	TXA	
80E0- 8D DE C0	1360	STA IER		873C- 48	2000	PHA	
	1370			873D- 98	2010	TYA	
80E3- 58	1380	CLI		873E- 48	2020	PHA	
80E4- 60	1390	RTS			2030		
80E5- 00 00	1400	.BY 0 0		873F- AD E9 80	2040 NEU	LDA AUS	;ein aus-
80E7- 00 00	1410 CSWSP	.BY 0 0		8742- D0 05	2050	BNE NEXT2	;zugebendes
80E9- 00 00	1420 FREI	.BY 0 0		8744- AD EA 80	2060	LDA AUS+1	;Zeichen
80EB- 00 00	1430 AUS	.BY 0 0		8747- F0 48	2070	BEQ END2	;vorhanden?
	1440 BUFFER	.DS 1535			2080		
86EA- 00	1450 BUFFEND	.BY 0		8749- 2C C1 C1	2090 NEXT2	BIT DTEST	;Drucker bereit?
	1470 ;			874C- 30 43	2100	BMI END2	
	1480 ;Speicher-Programm				2110		
	1490 ;				2120	;Zähler ändern	
86EB- 08	1500 SPEICH	PHP	;Register retten	874E- CE E9 80	2130	DEC AUS	;ein
86EC- 48	1510	PHA		8751- AD E9 80	2140	LDA AUS	;auszugebendes
	1520			8754- C9 FF	2150	CMP #FF	;Zeichen
86ED- AD E7 80	1530 WARTE		;ist	8756- D0 03	2160	BNE L5	;weniger
86FO- D0 05	1540	BNE NEXT1	;der	8758- CE EA 80	2170	DEC AUS+1	
86F2- AD E8 80	1550	LDA FREI+1	;Puffer		2180 L5		
86F5- F0 F6	1560	BEQ WARTE	;frei?	875B- EE E7 80	2190	INC FREI	;ein
	1570			875E- D0 03	2200	BNE LZEIG	;freier
	1580 ;Zähler ändern			8760- EE E8 80	2210	INC FREI+1	;Platz mehr


```

2220 LZEIG
8763- AD 00 00 2230 LDA #0
8766- BD 90 C0 2240 STA DRUCK
8769- C9 BD 2250 CMP #CR ;Druckerabhängig:
876B- D0 04 2260 BNE WEITER ; wenn Wagenrücklauf
876D- A9 8A 2270 LDA #LF ; dann auch Zeilenvorschub
876F- D0 F5 2280 BNE P
8771- AD 64 87 2290 WEITER LDA LZEIG+1
8774- C9 EA 2300 CMP #L,BUFFEND
8776- AD 65 87 2310 LDA LZEIG+2 ;Bufferende
8779- E9 86 2320 SBC #H,BUFFEND
877B- EE 64 87 2330 INC LZEIG+1 ;erreicht?
877E- D0 03 2340 BNE L7
8780- EE 65 87 2350 INC LZEIG+2
2360 L7
8783- 90 BA 2370 BCC NEU ;C=1 wenn Ende erreicht
8785- A9 EB 2380 LDA #L,BUFFER
8787- BD 64 87 2390 STA LZEIG+1 ;LZEIG auf
878A- A9 80 2400 LDA #H,BUFFER
878C- BD 65 87 2410 STA LZEIG+2 ;Anfang setzen
878F- B0 AE 2420 BCS NEU
2430
8791- AD DD C0 2440 END2 LDA IFR ;T1-Interrupt-Flag
8794- 29 40 2450 AND #%01000000
8796- BD DD C0 2460 STA IFR ;löschen
8799- 68 2470 PLA
879A- A8 2480 TAY
879B- 68 2490 PLA
879C- AA 2500 TAX
879D- AD 45 00 2510 LDA #45
87A0- 40 2520 RTI
2530
87A1- 4C 00 00 2540 EXIT JMP #0 ;Sprung zur alten INT.-Routine
2550
2560 .EN
    
```

unserer Routine fortgesetzt, die fast dasselbe tut, wie die „RDKEY“-Routine. Sie holt ebenfalls das nun ja blinkende Zeichen und wandelt es in ein „inverses“ Zeichen um. Danach erfolgt ein Sprung auf die Standard-Eingaberoutine „KEYIN“.

Erweitert man die Routine auf:
 300: 48 B1 28 29 3F 09 80
 91 28 68 4C 1B FD
 so verschwindet der Cursor ganz.

Ronald Pfitzer

Apfel-Menü

Der Apple-II verfügt über eine Eigenschaft, die ihn zum sogenannten „Turnkey“-System machen kann: Man legt nur eine Diskette ein, schaltet den Computer ein und schon wird ein Basic-Programm gestartet. Sinnvollerweise führt man dieses zuerst ausgeführte Basic-Programm als Menü aus, das eine Übersicht und eventuell kurze Erläuterung der auf dieser Disk zur Verfügung stehenden Programme gibt.

Das im Bild abgedruckte Programm erlaubt es außerdem, alle aufgeführten Programme mit einem einzigen Knopfdruck zu starten. Die Programme sind in der Reihenfolge Bezeichnung, Ladesequenz in DATA-Zeilen abgelegt (auch der Befehl CATALOG wurde hier so realisiert). Das Listing enthält einige Beispiele hierfür. Sinnvollerweise geht man bei der Initialisierung neuer Disketten so vor, daß man zuerst das Menüprogramm von einer anderen Disk lädt und dann INIT HELLO eingibt, so daß das Menü als HELLO-File dient.

Herwig Feichtinger

Nervenschonender Cursor für den Apple

Viele Bekannte, denen ich meinen Apple-II vorstellte, fragten mich, ob ich das „blinkende Ding“ nicht abstellen könnte. Zweifellos ist der blinkende Cursor nichts für labile Zeitgenossen, denen ich das folgende Kurzprogramm wärmstens empfehlen kann:

```

300: 48 B1 28 29 3F 91
      28 68 4C 1B FD
    
```

Aktivieren kann man das Programm folgendermaßen ([.] zus. für DOS):
 In Basic: POKE 56,0:POKE 57,3
 [:CALL 1002]

Im Monitor: 38: 00 03 [N 3EAG]

Desaktivieren kann man es durch folgende Maßnahmen:

- Durch einen Druck auf die Resetaste
- In Basic durch das Kommando: IN#0
- Im Monitor durch: 0 Ctrl-K

Und das steckt dahinter:
 Der blinkende Cursor beim Apple

kommt dadurch zustande, daß die Eingabe-Routine „RDKEY“ das aktuelle Zeichen vom Bildschirmspeicher holt und dann durch geeignete Bitmanipulation das zuvor „normale“ Zeichen in ein blinkendes umwandelt. Durch „verbiegen“ des Eingabevektors wird der Ablauf nach der Ausführung von „RDKEY“ bei

Menüprogramm für Apple-Disketten

```

10TEXT:SPEED=255:HOME:INVERSE
15PRINT" MC - DIE MIKROCOMPUTER-ZEITSCHRIFT "
20NORMAL:PRINT:PRINT"Bitte waehlen Sie:":PRINT
30READA$:IFA$="END"THEN50
40PRINTA$:READA$:GOTO30
50PRINT:GETA$:PRINTA$;" ";:RESTORE
60IFA$=CHR$(3)THENSTOP
70READB$:IFB$="END"THENRUN
80IFA$(<)>LEFT$(B$,1)THENREADB$:GOTO70
90READB$:PRINTB$:PRINTCHR$(4);B$
95GETA$:RUN
99REM PROGRAMMNAMEN UND LADESEQUENZEN
100DATA 1 Basicalc mc 12/1983 S.73,RUN BASICALC
110DATA 2 Memview mc 1/1984 S.57,BRUN MEMVIEW
950DATA,
960DATA G mc-Titelgrafik,RUNHELLO
970DATA F Programme kopieren,BRUNFID
980DATA D Diskette kopieren,BRUNSCOPY
990DATA C Disk-Inhaltsverzeichnis,CATALOG
999DATAEND
    
```

Rudolf Hofer

Apple liest CBM-Dateien

Die mc-Redaktion verwendet den CBM-3032 zusammen mit dem in [1] beschriebenen Basic-Editor, um ihr Jahresinhaltsverzeichnis zu erstellen. Der Autor dieses Beitrags wollte die vorhandenen Dateien auch mit dem für den Apple modifizierten Editor [2] lesen. Es entstanden zwei Programme, die CBM-Dateien vom Band lesen und auf Diskette übertragen. Die Daten können direkt vom Editor geladen werden.

Die Leseroutine ist in Bild 1 abgedruckt. Grundlage dafür ist ein Programm, das in [3] beschrieben wurde. Einzelheiten zum Kassettenformat des CBM und zur Byteseleroutine sind dort erläutert. Wichtig für den einwandfreien Betrieb ist vor allem ein sauberes Signal vom Kassettenrecorder – nachzuprüfen am einfachsten mit einem Oszilloskop. Das auf TTL-Pegel aufbereitete Signal kann von IC H14 (Anschluß 4) abgegriffen werden. Beim zehn Sekunden dauernden Vorspann dürfen keine Phasensprünge mehr auftreten.

Werden verschiedene Kassettenrecorder für Aufnahme und Wiedergabe verwendet, dann kann eine unterschiedliche Kopfeinstellung dazu führen, daß höhere Frequenzen nicht mehr mit ausreichendem Pegel wiedergegeben werden. Falls man an die Stellschraube herankommt, läßt sich die Wiedergabe leicht per Gehör (auf hellsten Klang) einstellen. Aber auch folgende Methode ist praktikabel: Das Originalband wird mit zwei Recordern überspielt, und zwar stark übersteuert. Ein weiteres Problem kann auftreten,

weil das Programm nicht einen Pegelwechsel, sondern positive oder negative Flanke erkennt. Beim Test sollte deshalb immer auch mit einer Version experimentiert werden, bei der die beiden Branch-Befehle an den Adressen 3AF und 3B5 vertauscht sind. Dazu muß man lediglich das Byte 10 mit dem Byte 30 austauschen. Falls das Programm irgendwann einmal aus unerklärlichen Gründen nicht mehr läuft, sollte man sich an diese Tatsache erinnern. Natürlich nur, wenn man nicht mit den Recordern gearbeitet hat, mit denen bereits ein Test erfolgreich war.

Wichtig für die einwandfreie Funktion ist auch, daß das Programm erst gestartet wird, wenn sich das Band im Vorspann befindet und der Recorder bereits seit ein bis zwei Sekunden eingeschaltet ist.

Basic-Programm erzeugt die Zieldatei

Das in Bild 2 wiedergegebene Basic-Programm ruft zunächst die Leseroutine auf, die natürlich vorher geladen werden muß. Danach steht die gelesene Datei ab 1000 (hex.) im Speicher. Jetzt fordert es den Bediener dazu auf, eine Diskette einzulegen, und nach Return überträgt es den Speicherinhalt in die Datei mit dem Namen File. Als letzter Eintrag wird „END“ angefügt, da dies für den mc-Editor als Abschluß notwendig ist.

0800	1	%CBM-BAENDER LESEN	0332	A505	38	LDA BLCNT	037C	B008	75	BCS LONG	
0800	2	ZP0 EPZ 0	0334	20DAFD	39	JSR PRBYT	037E	E028	76	CPX ##28	
0800	3	ZP1 EPZ 1	0337	A501	40	LDA ZP1	0380	B0F3	77	BCS BYTE	
0800	4	ZP2 EPZ 2	0339	C990	41	CMF ##90	0382	88	78	DEY	
0800	5	ZP3 EPZ 3	033B	F006	42	BEQ RTN	0383	10F2	79	BPL M3	
0800	6	ZP4 EPZ 4	033D	C605	43	DEC BLCNT	0385	60	80	RTS	
0800	7	BLCNT EPZ 5	033F	A505	44	LDA BLCNT	0386	A009	81	LONG LDY #7	
0800	8	CDUT EQU \$FDFO	0341	D0E7	45	BNE READ	0388	8403	82	STY ZP3	
0800	9	PRBYT EQU \$FDFA	0343	60	46	RTN RTS	038A	20A903	83	JSR LAENG	
0800	10	TAPEIN EQU \$C060	0344		47	-----	038D	209E03	84	M5 JSR BIT	
0800	11	PUFFER EQU \$1000	0344	A910	48	BLOCK LDA /PUFFER	0390	9002	85	RCC M4	
0800	12	-----	0346	B501	49	STA ZP1	0392	E603	86	INC ZP3	
0300	13	ORG \$300	0348	A900	50	LDA #PUFFER	0394	6A	87	M4 RDR	
0300	14	OBJ \$B00	034A	B500	51	STA ZP0	0395	88	88	DEY	
0300	204403	15	HEADER JSR BLOCK	034C	A910	52	NBLOCK LDA ##10	0396	D0F5	89	BNE M5
0303	AD0810	16	LDA PUFFER+##	034E	B504	53	STA ZP4	0398	2A	90	ROL
0306	B506	17	STA BLCNT+1	0350	207503	54	M9 JSR BYTE	0399	49FF	91	EOR ##FF
0308	B505	18	STA BLCNT	0353	3003	55	BMI M8	039B	4603	92	LSR ZP3
030A	A200	19	LDX #0	0355	4C4C03	56	JMP NBLOCK	039D	60	93	RTS
030C	BD0E10	20	M16 LDA PUFFER+##E,X	0358	C604	57	M8 DEC ZP4	039E		94	-----
030F	C920	21	CMF ##20	035A	D0F4	58	BNE M9	039E	20A903	95	BIT JSR LAENG
0311	F008	22	BEQ M15	035C	207503	59	M10 JSR BYTE	03A1	B602	96	STX ZP2
0313	C980	23	ORA ##80	035F	30FB	60	BMI M10	03A3	20A903	97	JSR LAENG
0315	20F0FD	24	JSR CDUT	0361	B004	61	M13 BCS M11	03A6	E402	98	CPX ZP2
0318	EB	25	INX	0363	A9FE	62	LDA ##FE	03A8	60	99	RTS
0319	D0F1	26	BNE M16	0365	E604	63	JNC ZF4	03A9		100	-----
031B	A2FF	27	M15 LDX ##FF	0367	9100	64	M11 STA (ZP0),Y	03A9	A200	101	LAENG LDX #0
031D	A0FF	28	M18 LDY ##FF	0369	E600	65	INC ZP0	03AB	E8	102	M6 INX
031F	88	29	M17 DEY	036B	D002	66	BNE M12	03AC	2C60C0	103	BIT TAPEIN
0320	D0FD	30	BNE M17	036D	E601	67	INC ZP1	03AF	10FA	104	BPL M6
0322	A910	31	LDA /PUFFER	036F	207503	68	M12 JSR BYTE	03B1	E8	105	M7 INX
0324	B501	32	STA ZP1	0372	10ED	69	BPL M13	03B2	2C60C0	106	BIT TAPEIN
0326	A900	33	LDA #PUFFER	0374	60	70	RTS	03B5	30FA	107	BMI M7
0328	B500	34	STA ZP0	0375		71	-----	03B7	60	108	RTS
032A	204C03	35	READ JSR NBLOCK	0375	A011	72	BYTE LDY ##11	03B8		109	-----
032D	A9BD	36	LDA ##BD	0377	20A903	73	M5 JSR LAENG	03B8		110	FAU
032F	20F0FD	37	JSR CDUT	037A	E03C	74	CPX ##3C				

Bild 1. Dieses Programm überträgt die CBM-Datei in einen Pufferbereich ab 1000 (hex); es wird mit CALL768 aufgerufen

Das Apple-Sonderheft

Soll eine Textdatei für andere Zwecke erstellt werden, kann Zeile 123 einfach entfallen. Der Text im Speicher wird nicht zerstört. Das erweist sich besonders beim Test als hilfreich. Man braucht nämlich die Leseroutine nur einmal aufzurufen und kann dann den Transferteil immer wieder mit RUN 5 starten.

Ein kleiner Schönheitsfehler soll nicht verschwiegen werden. Das Leseprogramm erkennt zwar einen Fehler und liest dann den jeweiligen Block noch einmal (der CBM zeichnet jeden Block zweimal auf), tritt aber auch da ein Fehler auf, dann wird das vom Programm nicht mehr erkannt.

Literatur

- [1] Feichtinger, Herwig: Auf der Suche nach Literatur. mc 1982, Heft 9, Seite 57.
- [2] Hofer, Rudolf: Apple auf Literatursuche. mc 1983, Heft 3.
- [3] Feichtinger, Herwig: AIM liest CBM-Kassetten. mc 1982, Heft 3, Seite 36.

```
1 HOME : PRINT "COM-BAND WIRD BELESEN": PRINT
2 PRINT "ANZEIGT WERDEN DATEI-NAME"
3 PRINT "UND ZAHL DER NOCH ZU LESENEN BLOCKE": PRINT
4 CALL 768
5 HIMEM: 4095: INPUT "INITIALISIERTE DISKETTE EINLEGEN (RET)":B$
20 BLOCKZAHL = PEEK (6)
25 D$ = CHR$ (4): PRINT D$"NON O": PRINT "$"OPEN FILE"
26 PRINT D$"WRITE FILE"
30 FOR I = 0 TO BLOCKZAHL - 2
35 FOR J = 10 TO 200
40 CH = PEEK (4096 + I * 203 + J)
42 IF CH = 13 THEN PRINT A$:A$ = "": GOTO 60
44 A$ = A$ + CHR$ (CH)
60 NEXT J
70 NEXT I
80 FOR J = 10 TO 200
90 CH = PEEK (4096 + I * 203 + J)
94 IF CH = 13 THEN PRINT A$:A$ = "": GOTO 115
100 IF CH = 0 THEN 120
105 A$ = A$ + CHR$ (CH)
115 NEXT J
120 IF A$ < > "" THEN PRINT A$
123 PRINT "END"
130 PRINT D$"CLOSE FILE"
```

Bild 2. Die Datei wird auf Diskette übertragen, sie kann unter dem Namen File vom mc-Editor aus aufgerufen werden

Vom Umgang mit Apple-Maschinenprogrammen

Immer wieder haben Nur-Basic-Programmierer Schwierigkeiten im Umgang mit fertigen Maschinenprogrammen, die lediglich einzutippen, abzuspeichern und wieder zu laden sind. Deshalb hier eine kurze Anleitung anhand des Beispiels in Bild 1.

Abgedruckt ist hier das sogenannte Assemblerlisting. Das bedeutet, neben dem eigentlichen Code, der in den Speicher zu bringen ist, wird noch zusätzliche Information dargestellt. Wichtig für den Benutzer sind die linken beiden Spalten. Sie enthalten: 1. die Adresse, 2. den Code. Wie man in Bild 1 sieht, wird in den ersten Zeilen kein Code erzeugt. Bis dahin ist auch die Adresse uninteressant. Los geht's dann mit den Code-Bytes 20 44 03 ab Adresse 300. Alles was in dieser Spalte steht, ist in den Computer einzugeben. Übrigens entspricht das dem sogenannten Hex-Dump oder Hex-Listing, das oft anstelle des Assemblerlistings (aus Platzgründen) abgedruckt wird. Beim Hex-Listing steht üblicherweise links eine Adresse, und rechts davon folgen acht oder 16 Code-Bytes. Das ergibt eine kompakte Blockdarstellung,

Bei der Eingabe geht man sinnvollerweise so vor:

- Mit CALL-151 kommt man in den Apple-Monitor, der sich mit einem Sternchen meldet.
- Jetzt tippt man die erste Adresse ein, gefolgt von einem Doppelpunkt und ca. zwei Zeilen Code (im Beispiel: 300: 20 44 03 AD 0B 10 usw.).
- Nach Return steht das ganze im Speicher.
- Die restlichen Daten werden auf dieselbe Weise eingegeben (also z. B.: 313: 09 80 20 F0 FD usw.).
- Kontrollieren kann man die eingegebenen Daten, indem man Anfangsadresse und Endadresse, getrennt durch einen Punkt, eingibt und mit Return abschließt (300.3B7). Der Monitor gibt daraufhin den gesamten Block aus.
- Hat man einen Fehler gemacht, korrigiert man die entsprechende Speicherstelle durch Eingabe von Adresse, Doppelpunkt und Code

wie gehabt. Natürlich ist die Adresse durch Weiterzählen im Hexadezimalcode erst zu ermitteln, wenn ein Byte geändert werden soll, das nicht unmittelbar der Adressenangabe folgt.

- Jetzt kann das Programm abgespeichert werden. Man gibt BSAVE Name, A\$ Anfangsadresse, L\$ Länge ein (im Beispiel BSAVE Name, A\$300, L\$B8). Es schadet nicht, wenn man die Angabe für die Länge zur Sicherheit etwas größer wählt. Alle Werte sind hexadezimal anzugeben.
- Eigentlich sind wir jetzt fertig. Das Programm kann jederzeit mit BLOAD Name wieder geladen werden, auch von einem Basic-Programm aus, wenn im Print-Befehl wie üblich CHR\$ (4) vorher ausgegeben wird. Mit BRUN Name kann man das Programm laden und gleich starten. Das funktioniert aber nur, wenn die Anfangsadresse gleichzeitig die Startadresse ist, wie im vorliegenden Fall.
- Ctrl-C bringt uns wieder in den Basic-Interpreter zurück.

Rudolf Hofer

Apple-II auf Literatursuche

In Heft 9/1982 hatten wir auf Seite 57 ein universelles zeilenorientiertes Editor-Programm in CBM-Basic veröffentlicht, das sich neben anderen Dateiverwaltungs-Aufgaben auch sehr gut zum Anlegen einer Literaturstellen-Datenbank eignet. Mit einigen Änderungen ist es auch für den Apple-II verwendbar. Der Einfachheit halber drucken wir das gesamte Programm hier in einer Apple-Fassung ab.

Bild 1 zeigt den Basic-Teil des Editorprogramms, dessen Befehlssatz ja schon in Heft 9/1982 ausführlich besprochen wurde. Die Änderungen gegenüber der CBM-Originalversion betreffen in erster Linie die Disketten-Befehle sowie die

Cursor-Steuerzeichen. Leider verfügt der Apple-II in seiner Normalversion nicht über die Möglichkeit der gemischten Eingabe von Groß- und Kleinbuchstaben, was aber bei einer Datenbank nur von sekundärem Interesse ist.

Für die schnelle Stichwortsuche verwendet der Basic-Teil eine Maschinenroutine, die mit dem neuen Befehl INST aufgerufen wird. Sie ist in Bild 2 als Hex-Dump aufgelistet und entspricht in ihrer Arbeitsweise derjenigen, die bereits für den CBM veröffentlicht wurde. Sie ist für die 48-KByte-Version des Apple-II ausgelegt (soviel Speicher sollte man für die Dateiverwaltung schon haben) und wird vom Apple-Monitor aus an der hexadezimalen Adresse 9400 gestartet. Dabei setzt sie automatisch die obere Basic-Speichergrenze (HIMEM) und schützt sich dadurch selbst vor dem Überschreiben mit Basic-Stringvariablen.

Damit das Programm z. B. bei Disketten-Lese Fehlern nicht abbricht, enthält Zeile 1 des Basic-Teils die Anweisung „ON ERR GOTO 450“. Dort wird dann eine Fehlermeldung ausgegeben, und das Programm ist wieder in der Befehls-eingabe-Schleife.

```

1 HOME : ONERR GOTO 450
2 PRINT "***** M C - E D I T O R *****"
4 D = 400:W = 80:S$ = "N": REM "NORMALWERTE F.ZEILENZAH, BR
   EITE,RUBRIKEN;V4.5/FE
8 DIM A$(D),B$(32),C$(100),N(32): DATA A,B,C,D,E,F,G,H,I,
   K,L,M,N,P,R,S,T,U,V,X,Y,Z
9 DS$ = CHR$(4)
10 GOSUB 238: GOSUB 432
12 INVERSE : PRINT "BEFEHL? ";
14 RESTORE : GET M$: IF M$ = "" GOTO 14
16 FOR I = 1 TO 22: READ B$: IF B$ = M$ THEN PRINT B$: NOR
   MAL : GOTO 20
18 NEXT : GOTO 14
20 IF I < 12 THEN ON I GOSUB 274,28,372,34,398,40,408,418,
   74,90,102: GOTO 24
22 ON I - 11 GOSUB 442,128,144,196,170,186,190,320,250,236,
   264
24 PRINT A$(J): GOTO 12
26 REM " *** BEFEHL B ***
28 FOR J = 0 TO D: IF A$(J) = "END" THEN RETURN
30 NEXT
32 REM " *** BEFEHL D ***
34 IF A$(J) < > "END" THEN J = J + 1
36 RETURN
38 REM " *** STICHWORT SUCHEN ***
40 INVERSE : J = - 1
42 INPUT "SUCHBEGRIFF=";B$:K = LEN (B$)
44 PRINT "DRUCKER? J/N": NORMAL
46 GET E$: IF E$ = "" THEN 46
48 IF E$ = "J" THEN GOSUB 312:Y = 1: PRINT B$: GOTO 52
50 Y = 0: IF E$ = "J" THEN RETURN
52 J = J + 1: IF A$(J) = "END" THEN RETURN
54 M$ = A$(J)
56 S = INST(M$,B$)
58 IF S = 0 THEN 52
59 IF Y = 1 THEN PRINT : PRINT DS$"PR#2"
60 PRINT M$
61 PRINT DS$"PR#0"
62 FOR S = 0 TO 10: NEXT
64 GET E$: IF E$ < > " " GOTO 52

```

```

66 INVERSE : PRINT "F=FORTSETZUNG,S=STOP ": NORMAL
68 GET E$: IF E$ = "S" THEN PRINT "STOP": RETURN
70 IF E$ = "F" THEN PRINT "OK": GOTO 52
72 GOTO 68
74 I = - 1: REM " *** INSERT ***
76 I = I + 1: IF A$(I) < > "END" GOTO 76
78 A$(I + 1) = A$(I):I = I - 1: IF I > = J GOTO 78
80 INPUT M$: IF M$ = "END" THEN I = J: GOTO 96
82 IF LEFT$(M$,1) = " " OR LEFT$(M$,1) = "?" THEN M$ =
   MID$(M$,2): GOTO 82
86 A$(J) = M$:J = J + 1: GOTO 74
88 REM " *** ZEILE LOESCHEN ***
90 INVERSE : PRINT "LOESCHEN? J/N":I = J: NORMAL
92 GET M$: IF M$ < "J" GOTO 92
94 IF M$ > "J" THEN RETURN
96 I = I + 1: IF A$(I - 1) = "END" THEN RETURN
98 A$(I - 1) = A$(I): GOTO 96
100 REM " *** FILE LADEN ***
102 INVERSE : IF A$(0) = "END" GOTO 112
104 PRINT "BISHERIGE TEXTE LOESCHEN? J/N";
106 GET M$: IF M$ = "" GOTO 106
108 INVERSE : PRINT M$: IF M$ = "J" THEN GOSUB 238
110 GOSUB 28
112 INPUT "FILENAME=";M$
118 PRINT DS$;"OPEN ";M$: PRINT DS$;"READ ";M$: PRINT "LADE
   VORGANG "
120 INPUT A$(J): IF ASC (A$(J)) < 33 THEN A$(J) = MID$(A
   $(J),2)
122 IF A$(J) < > "END" AND A$(J) < > "SEP" THEN J = J + 1
   : GOTO 120
124 A$(J) = "END": GOTO 182
126 REM " *** RUBRIKEN ERZEUGEN ***
128 IF A$(J) = "END" THEN RETURN
138 INVERSE : INPUT "RUBRIK";M$: NORMAL : IF LEN (M$) = 1
   THEN M$ = "#" + M$
140 A$(J) = M$ + " " + A$(J): GOTO 34
142 REM " *** AUSDRUCKEN "
144 INVERSE : PRINT "AUSDRUCK AB DER AKTUELLEN ZEILE": PRIN
   T "DRUCKER? J/N";R$ = ""
146 GET E$: IF E$ = "" GOTO 146

```

Bild 1. An den Apple-II-Befehlssatz angepaßte Version des mc-Editors aus Heft 9/1982 (Zeile 56: \$ = @)


```

148 PRINT E$: NORMAL
150 IF E$ = "J" THEN GOSUB 312: GOTO 154
152 IF E$ < > "N" THEN RETURN
154 FOR I = J TO D: IF A$(I) = "END" THEN RETURN
156 IF A$(I) > "" THEN GOSUB 208
158 PRINT DS$"PR#0": IF PEEK (- 16384) < = 127 THEN NEX
T I
160 INVERSE : PRINT "F=FORTSETZUNG,S=STOP": NORMAL
162 GET F$: IF F$ = "F" THEN NEXT I
164 IF F$ < > "S" GOTO 162
166 RETURN
168 REM " *** FILE ABSPEICHERN ***
170 B$ = "": IF A$(J) = "END" OR A$(J) = "SEP" THEN PRINT "
LEERES FILE": RETURN
172 FLASH : PRINT "GANZES FILE NUR NACH T": INPUT "FILENAME
=";M$: NORMAL : PRINT "ABSP.AB AKTUELLER ZEILE"
178 PRINT DS$;"OPEN ";M$: PRINT DS$;"WRITE ";M$
180 PRINT A$(J): IF A$(J) < > "END" AND A$(J) < > "SEP" T
HEN J = J + 1: GOTO 180
182 PRINT DS$;"CLOSE ";M$: RETURN
184 REM " *** BEFEHL T ***
186 J = 0: RETURN
188 REM " *** BEFEHL U ***
190 IF J > 0 THEN J = J - 1
192 RETURN
194 REM " *** RUBRIKFELD LOESCHEN ***
196 PRINT "RUBRIKFELD LOESCHEN? J/N": NORMAL
198 GET M$: IF M$ = "" GOTO 198
200 IF M$ < > "J" THEN RETURN
202 S$ = "N": PRINT "GELOESCHT"
204 FOR I = 0 TO D: IF A$(I) = "END" THEN RETURN
206 A$(I) = MID$ (A$(I),4): NEXT
208 M$ = A$(I):B$ = LEFT$ (M$,2): IF E$ = "J" THEN GOSUB 3
17
209 IF S$ = "N" GOTO 214
210 IF B$ < > R$ THEN R$ = B$: PRINT : PRINT R$
212 M$ = MID$ (M$,4)
214 IF LEFT$ (M$,1) = " " THEN M$ = MID$ (M$,2): GOTO 214
216 IF W > LEN (M$) THEN C$ = M$: GOTO 242
218 K = W + 1
220 K = K - 1: IF K = 0 THEN K = W: GOTO 228
222 B$ = MID$ (M$,K,1): IF B$ > "?" GOTO 220
224 IF B$ < ":" AND B$ > "." GOTO 220
226 IF B$ = "" OR B$ = "(" GOTO 220
228 C$ = LEFT$ (M$,K): GOSUB 242
230 M$ = MID$ (M$,K + 1): PRINT " ";
232 GOTO 214
234 REM " *** ALLES LOESCHEN ***
236 GOSUB 390: IF M$ < > "J" THEN RETURN
238 FOR I = 0 TO D:A$(I) = "": NEXT
240 J = 0:A$(J) = "END": PRINT "SPEICHER GELOESCHT": PRINT :
RETURN
242 D$ = "": PRINT C$: RETURN
248 REM " *** BEFEHL X ***
250 INVERSE : PRINT "ZUR ZEIT KEINE HILFSPROGRAMME": NORMAL
: RETURN
262 REM " *** BEFEHL Z ***
264 INVERSE : INPUT "DRUCKER-ZEILENBREITE";W
266 PRINT "RUBRIKFELDER? J/N ";
268 GET S$: IF S$ = "" GOTO 268
270 PRINT S$: NORMAL : RETURN
272 REM " *** SORTIEREN ***
274 INVERSE : PRINT "FILE WIRD SORTIERT": FOR I = 0 TO D: I
F A$(I) < > "END" THEN NEXT
276 I = I - 1:L = 1:N(L) = I + 1:M = 0
278 J = N(L):K = M - 1: IF J - M < 3 GOTO 300
280 M1 = INT ((K + J) / 2): IF PEEK (- 16384) > = 127 TH
EN PRINT "SORTIERVORGANG UNTERBROCHEN": NORMAL : RETURN
282 K = K + 1: IF K = J GOTO 292
284 IF A$(K) = < A$(M1) GOTO 282
286 J = J - 1: IF K = J GOTO 292
288 IF A$(J) > = A$(M1) GOTO 286
290 Y$ = A$(K):A$(K) = A$(J):A$(J) = Y$: GOTO 282
292 IF K > = M1 THEN K = K - 1
294 IF J = M1 GOTO 298
296 Y$ = A$(K):A$(K) = A$(M1):A$(M1) = Y$
298 L = L + 1:N(L) = K: GOTO 278
300 IF J - M < 2 GOTO 306
302 IF A$(M) < A$(M + 1) GOTO 306
304 Y$ = A$(M):A$(M) = A$(M + 1):A$(M + 1) = Y$
306 M = N(L) + 1:L = L - 1: IF L > 0 GOTO 278
308 PRINT "**** SORTIERT ****": NORMAL : RETURN
310 RETURN
312 PRINT : PRINT DS$"PR#2"
314 REM DRUCKPARAMETER
315 PRINT DS$"PR#0"
316 RETURN
317 PRINT : PRINT DS$"PR#2": RETURN
318 REM " *** DISK-VERZEICHNIS ***
320 INVERSE : PRINT DS$"CATALOG": NORMAL
328 RETURN
372 REM ***CLEAR FILE***
374 INVERSE : INPUT "FILENAME=";M$: NORMAL
378 IF M$ = "MC-ED" OR M$ = "PRINT" OR M$ = "INST" OR M$ =
"HELLO" THEN RETURN
380 PRINT DS$"DELETE ";M$: RETURN
390 INVERSE : PRINT "ALLES LOESCHEN? J/N";
392 GET M$: IF M$ = "" THEN 392
394 PRINT M$: NORMAL : RETURN
396 REM " *** ZEILE EDITIEREN ***
398 GOSUB 416: IF A$(J) = "END" THEN RETURN
400 M$ = A$(J):S = INST(M$,B$)
402 IF S = 0 THEN PRINT "- NICHT IN: ";: RETURN
403 IF S = 1 THEN A$(J) = D$ + MID$ (A$(J),S + LEN (B$)):
RETURN
404 A$(J) = LEFT$ (A$(J),S - 1) + D$ + MID$ (A$(J),S + LE
N (B$))
406 RETURN
408 J = 0: PRINT "*** GLOBALE AENDERUNG ***": GOSUB 416
412 IF A$(J) = "END" THEN RETURN
413 GOSUB 400: PRINT A$(J):J = J + 1
414 GOTO 412
416 INVERSE : INPUT "ALTER STRING";B$: INPUT "NEUER STRING"
;D$: NORMAL : RETURN
418 RESTORE : PRINT "BEFEHLE:
420 FOR I = 0 TO 21: READ M$: NEXT
422 HOME : FOR I = 0 TO 10: READ M$: PRINT M$;: PRINT TAB(
21): READ M$: PRINT M$: NEXT
424 RETURN
428 DATA "A SORTIEREN","B LETZTE ZEILE","C CLEAR FILE","D
NAECHSTE ZEILE
430 DATA "E EDITIEREN","F FINDE WORT","G GLOBALE AENDERUNG"
,"H HILFE(LISTE)
432 DATA "I INSERT(EINGABE)","K ZEILE LOESCHEN","L FILE LAD
EN
433 DATA "M MEMORY-STATUS
434 DATA "N RUBRIZIEREN","P AUSDRUCKEN","R RUBRIKEN LOESCHE
N
436 DATA "S FILE ABSPEICHERN","T ERSTE ZEILE","U VORHERIGE
ZEILE
438 DATA "V DISK-INHALT","X DISK-HILFSPGM.
440 DATA "Y ALLES LOESCHEN","Z PARAM.AENDERN:?:?"," "
441 REM " *** STATUSABFRAGE ***
442 PRINT "- AUSDRUCK "W" ZEICHEN BREIT";
444 IF S$ = "J" THEN PRINT " , RUBRIZIERT";
446 GOSUB 28: PRINT : PRINT "- "J" ZEILEN (MAX."D")
448 PRINT "- FREIER SPEICHERPLATZ:" FRE (0)"BYTES": RETURN
450 INVERSE : PRINT "FEHLER": NORMAL :M$ = "": GOTO 12

```

```

9400- A9 00 85 73 A9 94 85 74
9408- A9 4C 85 BA A9 15 85 BB
9410- A9 94 85 BC 60 C9 40 F0
9418- 24 C9 3A B0 F7 4C BE 00
9420- E6 BB D0 02 E6 B9 60 A5
9428- BB D0 02 C6 E9 C6 BB 60
9430- 20 20 94 D0 03 20 27 94
9438- A0 00 B1 BB 60 A4 B9 C0
9440- 02 F0 D6 A9 00 85 1F B5
9448- 1B 20 35 94 C9 D0 F0 03
9450- 4C 12 D4 20 6A DD A5 12
9458- 48 A5 83 85 96 A5 B4 85
9460- 97 20 20 94 A0 05 20 B1
9468- 00 D9 6B 95 D0 E2 8B D0
9470- F5 20 30 94 20 E3 DF 20
9478- 69 D0 20 52 95 A0 00 B1
    
```

```

9480- 83 85 88 C8 B1 83 8D ED
9488- 94 8D 10 95 C8 B1 83 8D
9490- EE 94 8D 11 95 20 BE DE
9498- 20 E3 DF 20 69 DD 20 52
94A0- 95 A0 00 B1 83 85 92 C8
94A8- B1 83 BD EA 94 8D 0B 95
94B0- C8 B1 83 8D EB 94 8D 0C
94B8- 95 20 38 94 C9 29 F0 27
94C0- C9 2C F0 03 4C 12 D4 20
94C8- 20 94 20 67 DD 20 6A DD
94D0- 20 0C E1 A5 A1 85 1F C5
94D8- 88 90 02 B0 3C 20 38 94
94E0- C9 29 F0 03 4C 12 D4 A6
94E8- 1F AD 8D 93 DD 5C 93 F0
94F0- 09 EB E4 88 D0 F3 A2 00
94F8- F0 1D 86 1A A9 00 85 1C
    
```

```

9500- E6 1A E6 1C A4 1C C4 92
9508- F0 0C B9 8D 93 A4 1A D9
9510- 5C 93 D0 DD F0 EA EB 86
9518- 1B A5 96 85 83 A5 97 85
9520- 84 68 10 0E A0 00 A9 00
9528- 91 83 C8 A5 1B 91 83 4C
9530- 47 95 A9 00 85 9E A4 1B
9538- 84 9F A2 90 38 20 01 E3
9540- A8 83 A4 84 20 2B EB 68
9548- 68 68 68 20 20 94 4C D5
9550- D7 60 E0 00 D0 FB A2 12
9558- BD 70 95 20 5C DB CA D0
9560- F7 A0 28 20 19 ED 20 DD
9568- FB 4C 03 E0 28 54 53 4E
9570- 49 44 4E 55 4F 46 20 54
9578- 4F 4E 20 47 4E 49 52 54
9580- 53 3F 00
    
```

Bild 2. Maschinenprogramm zur schnellen Stichwortsuche (Aktivierung mit 9400 G oder BRUN INST)

ASCII-Zeichenfolgen sichtbar gemacht

Wenn man fremde Maschinenprogramme analysiert oder englischsprachige Meldungen durch deutsche ersetzen möchte, so ist das mit einem normalen Monitor-Befehl, der den Speicherinhalt nur hexadezimal auflistet, recht mühsam. Denn erst einmal muß man ja herausfinden, in welchen Adressenbereichen überhaupt Texte stehen.

Das im Bild abgedruckte Apple-Programm löst dieses Problem. Nach dem Start mit 8400G vom Monitor aus ist der zusätzliche Monitorbefehl CTRL-Y aktiviert, der Speicherinhalte als ASCII-Zeichen darstellt. Ein Beispiel: Angezeigt werden soll der Speicherbereich ab 8440. Man gibt also im Monitor nach

```

8400- A9 10 8D F9 03 A9 84 8D
8408- FA 03 20 42 84 4C 69 FF
8410- A6 3C A4 3D 20 96 FD 20
8418- 48 F9 A2 10 A0 00 B1 3C
8420- E6 3C D0 02 E6 3D 09 80
8428- C9 A0 B0 02 A9 AE 20 F0
8430- FD CA D0 E8 20 0C FD C9
8438- A0 F0 01 60 4C 10 84 EA
8440- EA EA A2 00 BD 4F 84 F0
8448- F2 20 F0 FD E8 D0 F5 8D
8450- C3 D4 D2 CC AD D9 A0 C1
8458- C3 D4 C9 D6 C5 8D 00 00
    
```

Beliebige Speicherbereiche lassen sich mit diesem Apple-II-Programm als ASCII-Zeichen darstellen

dem Prompt-Sternchen ein: 8440 CTRL-Y, gefolgt von Return. Jetzt werden 16 Speicherzellen angezeigt. Um die nächsten 16 sichtbar zu machen, genügt ein Druck auf die Leertaste. Jede andere Taste führt wieder zur Anzeige des Monitor-Prompts.

Apple-Textdateien ohne OPEN lesen

Im DOS-Handbuch zum Apple-II ist erläutert, wie man eine Textdatei von Diskette in Strings einliest: nämlich mit der Kommandofolge OPEN Name, READ Name, INPUT String, CLOSE Name.

Dieses Vorgehen hat einen großen Haken: Wenn man auf diese Weise versucht, ein File zu lesen, was es gar nicht auf der Diskette gibt, so erscheint beim nächsten CATALOG der falsche Dateiname als neues File, ohne daß in ihm irgend etwas gespeichert wäre.

Nun probieren Sie mal folgendes aus: Lesen Sie Textfiles, indem Sie das OPEN am Anfang einfach weglassen. Das Ergebnis ist verblüffend: Entgegen den Angaben im DOS-Handbuch funktioniert das nämlich bestens, und es werden bei Angabe falscher Filenamen auch keine

Zur Anzeige werden alle gelesenen Bytes mit hex 80 oder-verknüpft, da der Apple ASCII-Zeichen nur bei gesetztem höchstwertigen Bit normal verarbeitet. Bytes, die nicht darstellbaren CTRL-Zeichen entsprechen, werden der Übersichtlichkeit halber durch einen Punkt ersetzt.

Ein Abspeichern des Programms ist mit BSAVE MEMVIEW, A\$8400, L\$60 auf Disk möglich. Der Start kann dann auch einfach durch BRUN MEMVIEW erfolgen.

Herwig Feichtinger

leeren Dateien mehr auf der Diskette erzeugt.

Die Befehlsfolge in Applesoft unter DOS 3.3 lautet also explizit:

```

100 D$=CHR$(4):INPUT"FILENAME:
";N$
110 PRINT D$;"READ"; N$:I=0
120 INPUTA$(I):IF
    A$(I)="END"THEN140
130 I=I+1:GOTO 120
140 PRINT D$;"CLOSE"
    
```

Diese kurze Routine liest eine Textdatei in das Feld A\$(I) bis zum letzten Element "END" (vgl. Literatursuche-Programm in mc 3/1983). Erwähnenswert ist noch, daß CLOSE nicht unbedingt mit einem Filenamen ergänzt werden muß.

Fe.

Wolfgang Stahn

Apple-II lernt sprechen

Bei der Sprachsynthese hat man normalerweise die Wahl zwischen Verfahren, die viel Speicher und wenig Zusatzaufwand oder wenig Speicher und viel Zusatzaufwand erfordern. Um so erstaunlicher ist es, daß man mit dem hier vorgestellten Miniprogramm ohne irgendwelche Zusatzhardware den Apple-II zum Sprechen bringen kann. Speicherbedarf: 1,2 KByte/s. Das Ergebnis klingt zwar etwas krächzend, ist aber erstaunlich gut verständlich.

Eingabe über Kassetteninterface

Die Eingangsschaltung des Kassetteninterface verlangt ein Signal mit einer maximalen Spannung von 1 V (Spitze-Spitze). Normalerweise liefert dieses Signal der Ohrhörerausgang des Kassettenrecorders, von dem man ein Programm einliest. Immer wenn die Wechselspannung am Eingang vom positiven in den negativen Bereich wandert (oder umgekehrt), ändert eine Flipflop-Schaltung ihren Zustand. Der Zustand dieses Flipflops (gesetzt oder nicht gesetzt) kann nun softwaremäßig über eine bestimmte Speicherstelle (C060 hex. bzw. 49248 dez.) abgefragt werden. Immer wenn der Wert dieser Adresse größer oder gleich 128 ist, ist das Flipflop gesetzt; ist er kleiner als 128, ist es nicht gesetzt. Leider arbeitet bei den meisten anderen Tischcomputern das Kassetteninterface nach einem anderen Prinzip, und das hier dargestellte Verfahren zur Sprachsynthese läuft darauf nur mit einigen Änderungen.

Legt man an den Eingang des Apple nun beispielsweise ein Sprachsignal, etwa aus dem Kopfhörerausgang eines auf „Aufnahme“ geschalteten Kassettenrecorders, in dessen Mikrofon man spricht, so ändert sich der Zustand des Flipflops fortwährend im Rhythmus des Eingangssignals. Wird jetzt per Programm der Zustand in sehr kurzen Abständen hintereinander abgefragt und abgespeichert, so wird die Sprachinformation in einer sehr groben Weise digital gespeichert (1-Bit-A/D-Umsetzung).

Ausgabe über den eingebauten Lautsprecher

Für die Wiedergabe der so gespeicherten Information verwendet man zweckmäßigerweise den eingebauten Lautsprecher. Auch er wird über eine bestimmte Adresse angesprochen (C030 hex. oder 49200 dez.). Bei jedem Zugriff auf diese Adresse ändert das Flipflop, das über einen Verstärker den Lautsprecher ansteuert, seinen Zustand. Beim Zugriff auf diese Adresse ist jedoch zu beachten, daß nur Lesebefehle (z. B. PEEK in Basic oder LDA in Maschinensprache) genau einen Pegelwechsel bewirken, während Schreibbefehle (z. B. POKE in Basic oder STA in Maschinensprache) zwei kurzzeitig aufeinanderfolgende Pegelwechsel produzieren. Soll die digital gespeicherte Sprache über den Lautsprecher wiedergegeben werden, muß man genau umgekehrt wie bei der Aufzeichnung vorgehen:

Die gespeicherte Information muß mit genau der gleichen Geschwindigkeit wie bei der Aufzeichnung aus dem Speicher gelesen werden. An den Stellen, an denen bei der Aufnahme das Eingangs-Flipflop seinen Zustand änderte, muß auf die Adresse des Lautsprecher-Flipflops zugegriffen werden, um dessen Ausgang umzuschalten. Hat man alles richtig gemacht, ist die gesprochene Information wieder zu hören. Die zweckmäßige Aufzeichnungsgeschwindigkeit liegt bei etwa 10 000 Abfragen des Eingangs-Flipflops in jeder Sekunde. Dies entspricht einem Datenfluß von 10 000 Bit/s. Der Speicherbedarf liegt somit bei rund 1,2 KByte/Sprechsekunde. Dies

reicht bei einem voll ausgebauten Apple für über eine halbe Minute synthetische Sprache!

Will man eigene Programme mit Sprachausgabe versehen, braucht man meistens sowieso nur wenige Worte oder Ziffern.

Ein Maschinenprogramm für Aufnahme und Wiedergabe

Das Programm (Bild 1) besteht im wesentlichen aus zwei Teilen: In Zeile 300 beginnt das Programmsegment, das für die Aufzeichnung von Sprachinformation erforderlich ist. In der assemblierten Form beginnt es im Speicher bei 800 hex. (2048 dez.). Das ist auch die Startadresse des Programms, um ein Sprachsignal abzuspeichern. Es ist zu beachten, daß die Konstanten FRPAGE, TOPAGE und VERZG, entsprechend den Adressen FD hex., FE hex. und FF hex. vor dem Programmstart vorzubesetzen sind. FRPAGE gibt die Speicherseite an, ab der die digitalisierte Information abgelegt werden soll. Hat man außer dem Grundprogramm keine weiteren Programme im Speicher, kann FRPAGE auf 0A hex. gesetzt werden. Dies bedeutet, daß das Programm die Information beginnend bei Adresse A00 hex. ablegt. TOPAGE gibt die Adressenseite an, von der ab keine Informationen mehr abgelegt werden darf, weil entweder kein RAM-Bereich mehr vorhanden ist oder wichtige Programme (z. B. DOS) sonst zerstört würden. Bei einem 48-K-Apple, bei dem der gesamte RAM-Bereich ausgenutzt werden kann, darf diese Konstante maximal den Wert C0 hex. erhalten. Dies bedeutet, daß die letzte Adresse, in der Information abgelegt wird, BFFF hex. ist.

Mit der Konstanten VERZG wird schließlich die Abfragerate festgelegt. Wird VERZG mit 1 vorbesetzt, so ergibt dies eine Geschwindigkeit von ca. 15 000 Abfragen in jeder Sekunde. Das bedeutet bestmögliche Sprachqualität bei größtem Speicherbedarf. Eine Vorbesetzung mit FF hex. ergibt zwar geringsten Speicherbedarf, doch nur eine Abtastrate von 700 Abfragen pro Sekunde. Hiermit läßt sich natürlich keine Sprachsynthese mehr betreiben. Der optimale Wert für VERZG liegt bei etwa 8. Damit wird auch die Abtastrate von 10 000 Abfragen pro Sekunde in etwa erreicht.

Der zweite Programmteil beginnt in Zeile 1080 und in der assemblierten Form bei Adresse 900 hex. (2304 dez.). Das ist die Startadresse, um die im Speicher befindliche Information wieder auszugeben. Die Konstanten, die vor dem Start

vorbesetzt werden müssen, sind die gleichen wie im ersten Programmsegment. Man muß jedoch aufpassen, daß bei der Wiedergabe zumindest VERZG den gleichen Wert wie bei der Aufzeichnung hat, da sonst die Sprache verzerrt wiedergegeben wird.

Startet man das Programm (Bild 2) bei 800 hex., will man also ein Sprachsignal digitalisieren und speichern, so springt das Programm zunächst in eine Warteschleife, in der auf einen mehrmaligen Wechsel des Zustandes des Eingangs-Flipflops gewartet wird. Das heißt, das Programm beginnt erst mit der Abspeicherung, wenn tatsächlich gesprochen wird. Es ist nicht sinnvoll, den Wert für die Zahl der Zustandswechsel, die abgewartet werden sollen, auf 1 zu setzen, da das Programm dann bei jedem kleinsten Knackimpuls losläuft. Gute Erfahrungen werden mit einem Wert von 5 gemacht. Zum Format der Abspeicherung ist noch folgendes zu bemerken:

*800.87D

```
0800- A5 FD 85 FC A9 00 85 FB
0808- A2 00 A9 00 81 FB 18 A5
0810- FB 69 01 85 FB A5 FC 69
0818- 00 85 FC C5 FE D0 EB A5
0820- FD 85 FC 20 D0 FB A2 05
0828- AD 60 C0 29 80 85 FA AD
0830- 60 C0 29 80 C5 FA 85 FA
0838- F0 F5 CA D0 F2 A9 01 85
0840- F9 A5 FD 85 FC AD 60 C0
0848- 30 05 A9 00 4C 54 08 A5
0850- F9 4C 54 08 01 FB 81 FB
0858- 20 7E 08 18 A5 FB 69 01
0860- 85 FB A5 FC 69 00 85 FC
0868- C5 FE D0 D9 A5 F9 C9 80
0870- F0 06 0A 85 F9 4C 41 08
0878- 20 D0 FB 4C 69 FF
```

Bild 2. Hex-Listing des Sprachsynthese-Programms. Ab 087B sollte allerdings besser hex 60 stehen

Der „Bitstrom“ wird nicht hintereinander im Speicher abgelegt. Hierzu wären zu viele Shift- und Maskierungsoperationen notwendig, die letztendlich zu

viel Zeit kosten würden. Das Programm legt die Bitfolge beginnend bei Bit 0 aller Bytes. Dann folgen Bit 1, Bit 2 usw. bis Bit 7. Das Programm durchläuft also insgesamt 8mal den für die Abspeicherung des Sprachsignals vorgesehenen Speicherbereich von Anfang bis Ende. Ist die Aufzeichnung oder Wiedergabe beendet, springt es zur Monitoradresse MONZ (FF69 hex.). Die beiden Sprünge in den Zeilen 990 und 1430 können auch gegen einen einfachen RTS-Befehl ausgetauscht werden. Dies ist insbesondere dann notwendig, wenn das Programm als Unterprogramm verwendet werden soll.

Grundlage für die Spracherkennung

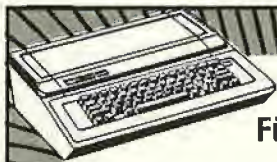
Basierend auf diesem Programm lassen sich auch Routinen zur Spracherkennung entwickeln. Eines meiner Experimentalprogramme funktioniert bei-

```
0080 ;+++ EIGENE VARIABLEN +++
0090 ;
0100 VERZG EQU FF
0110 TOPAGE EQU FE
0120 FRPAGE EQU FD
0130 POINTH EQU FC
0140 POINTL EQU FB
0150 LAST EQU FA
0160 MASKE EQU F9
0170 ;
0180 ;+++ I/O-ADRESSEN +++
0190 ;
0200 SPEAK EQU C030
0210 CASSIN EQU C060
0220 ;
0230 ; +++ MONITOR-ADRESSEN +++
0240 ;
0250 BELL EQU FBDD
0260 MONZ EQU FF69
0270 ;
0280 ORG 0800
0290 ;
0300 ; *** SPRACHEINGABE ***
0310 ;
0320 ; INITIALISIERUNG
0330 ;
0340 LDA FRPAGE
0350 STA POINTH
0360 LDA #00
0370 STA POINTL
0380 CLEAR LDX #00
0390 LDA #00
0400 STA (POINTL,X)
0410 CLC
0420 LDA POINTL
0430 ADC #01
0440 STA POINTL
0450 LDA POINTH
0460 ADC #00
0470 STA POINTH
0480 CMP TOPAGE
0490 BNE CLEAR
0500 LDA FRPAGE
0510 STA POINTH
0520 JSR BELL
0530 ;

0540 ; WARTEN AUF GERÄUSCH
0550 ;
0560 LDX #05
0570 LDA CASSIN
0580 AND #08
0590 STA LAST
0600 LOOP LDA CASSIN
0610 AND #08
0620 CMP LAST
0630 STA LAST
0640 BEQ LOOP
0650 DEX
0660 BNE LOOP
0670 ;
0680 ; AUFZEICHNUNG
0690 ;
0700 LDA #01
0710 STA MASKE
0720 LDA FRPAGE
0730 STA POINTH
0740 STORE LDA CASSIN
0750 BMI NEG
0760 LDA #00
0770 JMP ON
0780 NEG LDA MASKE
0790 JMP ON
0800 ON ORA (POINTL,X)
0810 STA (POINTL,X)
0820 JSR WAIT
0830 CLC
0840 LDA POINTL
0850 ADC #01
0860 STA POINTL
0870 LDA POINTH
0880 ADC #00
0890 STA POINTH
0900 CMP TOPAGE
0910 BNE STORE
0920 LDA MASKE
0930 CMP #08
0940 BEQ END
0950 ASL
0960 STA MASKE
0970 JMP INIT
0980 END JSR BELL
0990 JMP MONZ

1000 ;
1010 WAIT LDX VERZG
1020 WLOOP DEX
1030 BNE WLOOP
1040 RTS
1050 ;
1060 ORG 0900
1070 ;
1080 ; *** SPRACHAUSGABE ***
1090 ;
1100 LDA #01
1110 STA MASKE
1120 LDX #00
1130 LDA #00
1140 STA POINTL
1150 INIT2 LDA FRPAGE
1160 STA POINTH
1170 VOICE LDA (POINTL,X)
1180 AND MASKE
1190 CMP LAST
1200 STA LAST
1210 BEQ SKIP
1220 BIT SPEAK
1230 JMP NEXT
1240 SKIP NOP
1250 NOP
1260 JMP NEXT
1270 NEXT JSR WAIT
1280 CLC
1290 LDA POINTL
1300 ADC #01
1310 STA POINTL
1320 LDA POINTH
1330 ADC #00
1340 STA POINTH
1350 CMP TOPAGE
1360 BNE VOICE
1370 LDA MASKE
1380 CMP #08
1390 BEQ END2
1400 ASL
1410 STA MASKE
1420 JMP INIT2
1430 END2 JMP MONZ
?
```

Bild 1. Ohne Zusatzhardware ermöglicht dieses Programm, hier im Quellcode, die Sprachsynthese mit dem Apple-II



Für Apple II, II e

Z-80-Karte	99,— DM	80 Zeichen-Karte	159,— DM
Disk-Interface	109,— DM	(Autoswitch)	
PAL-Karte	109,— DM	Centronics-Interfacel	129,— DM
16 K-RAM-Karte	109,— DM	(mit Kabel für EPSON)	
RS 232-Karte	119,— DM	6809-Karte	399,— DM
Eprommer (4, 8, 16 K)	149,— DM	8088-Karte	759,— DM
128 K-RAM-Karte	488,— DM	Speech-Karte	88,— DM
AD-DA-Karte	119,— DM	Apple-Info 1,— DM (Porto)	
Wild-Karte	139,— DM		
(knackt geschützte Programme)			

Händleranfragen
erwünscht

Die ... Kompatiblen

Komp 48 995,— DM
48 K, 6502 ohne Firmware

Komp 64 1180,—
64 K, 6502, Z-80, 15er-Block
ohne Firmware

Komp 64 S 1330,— DM
wie Komp 64, jedoch mit abgesetzter
Tastatur mit 188 Funktionen.

Motherboard 48 K 499,— DM
8 Slots, alle IC's gesockelt
ohne Firmware, fertig geprüft

Motherboard 64 K 639,— DM
wie oben, mit 6502 und
Z80, 64 K

Klaus Jeschke
Hard-, Software
Im Birkenfeld 3a
6233 Kelkheim
☎ (06198) 7523

Alle Preise inklusive Mehrwertsteuer. 6 Monate
Garantie Versand erfolgt per NN oder Vorkasse

spielsweise folgendermaßen: Ein gesprochenes Wort wird mit Hilfe des Grundprogramms digitalisiert und in einem bestimmten Speicherbereich abgelegt. Anschließend wird eine Unterteilung der digitalen Information in 16 Segmente vorgenommen und in jedem dieser Segmente die Anzahl der Zustandswechsel gezählt. So erhält man für jedes Wort einen Satz von 16 Zahlen, die zusammen mit dem dazugehörigen Wort gespeichert werden können. Soll nun ein Wort erkannt werden, so wird es zunächst genauso behandelt. Anschließend werden für alle 16 Werte dieses Wortes die absoluten Differenzen zu den entsprechenden Zahlen eines jeden der bereits bekannten Wörter gebildet und addiert. Das Wort, bei dem diese Differenzensumme am kleinsten ist, entspricht also am meisten dem neu gesprochenen Wort. Die Erkennungsquote bei einem Satzsatz von 10 Worten lag beim gleichen Sprecher bei etwa 80 %. Dies soll nur eine Möglichkeit der Anwendung darstellen. Der Leser findet bestimmt noch mehr. Viel Spaß beim Experimentieren.

Literatur

[1] Feichtinger, Herwig: Sprache aus dem Computer. Mikrocomputer-Anwendungen, Sonderheft Nr. 33, Franzis-Verlag, S. 71...73.

Centronics — ganz einfach

Eine Centronics-Druckerschnittstelle läßt sich per Software sehr einfach realisieren. Dazu sind lediglich neun E/A-Leitungen erforderlich, wie Bild 1 zeigt. Zusammen mit Masse genügt also ein

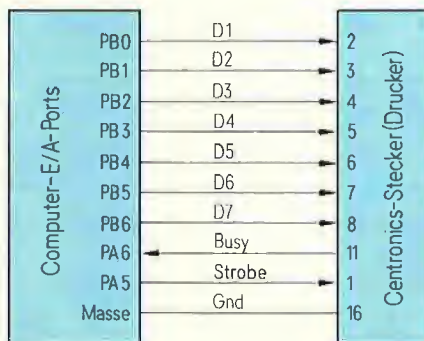


Bild 1. Verbindung des Centronics-Steckers mit zwei E/A-Ports

zehnnadriges Flachbandkabel. Die Treiber-Software in Bild 2 ist in 6502-Assembler auf einem Apple-II für eine

6532-Karte (Fa. Neucom, vgl. Modem-Programm aus mc 1/1984) in Slot 2 ausgelegt. Das Programm liegt bei ihr im 6532-RAM ab Adresse C200. Das Einschalten des Druckers erfolgt in Basic mit dem Befehl PR#2, im Monitor mit 2 CTRL-P und von einem Maschinen-sprache-Programm mit JSR \$C200.

Wieder ausschalten kann man den Drucker am Apple mit PR#0 bzw. innerhalb eines Programms unter DOS mit PRINT CHR\$(4); "PR#0". Im Monitor gibt man dazu 0 CTRL-P ein; Reset tut's aber in jedem Fall auch.

Das Programm verwendet nur die wichtigsten Schnittstellen-Leitungen des Centronics-Steckers.

Herwig Feichtinger

Bild 2.
6502-
Programm,
hier für eine
6532-Karte
im Apple-
Slot 2

```

0800      1  ;CENTRONICS-AUSGABEROUTINE
0800      2  ;PB0-6=D1-D7, PA5=STROBE, PA6=BUSY
0800      3  PA      EQU $C280      ;RIOT-
0800      4  PAD      EQU $C281      ;ADRESSEN
0800      5  PB      EQU $C282      ;FUER
0800      6  PBD      EQU $C283      ;SLOT 2
0800      7  OUTV     EPZ $36        ;AUSG.-VEKTOR
0800      8  VIDOUT    EQU $FDF0      ;VIDEOAUSG.
C200      9  ORG $C200              ;SLOT 2
C200 A91A     10  PR      LDA #C200      ;VEKTOR
C202 8536     11          STA OUTV      ;SETZEN
C204 A9C2     12          LDA /PRI
C206 8537     13          STA OUTV+1
C208 A97F     14          LDA #$7F      ;PB 0-6=
C20A 8D83C2   15          STA PBD      ;AUSGANG
C20D A920     16          LDA #$20      ;DUMMY-
C20F 8D82C2   17          STA PB       ;ZEICHEN
C212 8D81C2   18          STA PAD      ;PA 5=AUSG.
C215 8D80C2   19          STA PA       ;STROBE=1
C218 A98D     20          LDA #$8D      ;CR
C21A 2C80C2   21  PRI    BIT PA        ;BUSY?
C21D 70FB     22          BVS PRI      ;JA
C21F 8D82C2   23          STA PB       ;AUSGEBEN
C222 48       24          PHA          ;Z. RETTEN
C223 A900     25          LDA #0        ;STROBE=
C225 8D80C2   26          STA PA       ;LOW
C228 A9FF     27          LDA #$FF
C22A 8D80C2   28          STA PA       ;HIGH
C22D 68       29          PLA
C22E 4CF0FD   30          JMP VIDOUT
                31          END
    
```

Karl-Hermann Rollke

Apple-II liest und druckt Strichcode

In [1] stellten wir die Lösung dieses Problems in Basic und Maschinensprache vor. Jetzt zeigen wir Ihnen, wie die Aufgabe mit einem Pascal-Programm zu bewältigen ist. Die dabei verwendeten Prozeduren könnte man zum Beispiel in ein Bibliotheksprogramm einbauen, um Strichcodeaufkleber für Bücher zu drucken und sie per Lesestift zu erfassen.

Der hier gewählte Code besteht aus doppelt so breiten Einheiten, wie sie in [1] beschrieben wurden. D. h. Bit 1 entspricht sechs Strichen, Bit 0 zwei Strichen, und eine Lücke entspricht vier Strichen. Der Grund liegt darin, daß bei normaler Bewegung des Lesers in Pascal die Zählschleife leicht zu klein gerät. Durch einfache Modifikation läßt sich natürlich auch der „Standard-Code“ erzeugen und (mit etwas erhöhter Fehler-rate) lesen. Eine elegantere Modifikation

wäre allerdings ein Assemblerprogramm, das die Leseroutine ersetzt. Das Programm druckt 16-Bit-Zahlen, die von zwei Synchronisationsbits angeführt werden. Außerdem wird eine 8-Bit-Prüfsumme gedruckt, die ganz schlicht und einfach die Quersumme der ganzen Zahl darstellt. Da im Programm Integervariablen verwendet werden, müssen die darzustellenden Zahlen kleiner 32768 sein (Grund: Rechnen mit dem MOD-Befehl).

Funktioniert mit verschiedenen Druckern und dem mc-Strichcodeleser

Bei dem verwendeten Drucker handelt es sich um einen Epson FX-80. Das Programm funktioniert ebenfalls mit allen anderen grafikfähigen Epson-Druckern (MX-82, MX-80 mit Grafikoption, RX-80) sowie nach leichter Modifikation (Umkehrung der Bits 0 und 1) auf dem NEC PC-8023, bzw. ITOH 8510-A. Benutzt wird der in [2] beschriebene Leser, der entsprechend der Anleitung angeschlossen ist. D. h. sein Ausgang liegt am Eingang von „Pushbutton 0“ des Paddle-Anschlusses. Daher wird im Programm die Unit „Applestuff“ verwendet, damit die Funktion BUTTON(0) zugänglich ist.

Das Programm (Bild 1) besteht aus dem Druckteil und dem Leseteil. Will man einen Strichcodeausdruck (Bild 2) erzeugen, so wird zuerst nach der zu druckenden Zahl gefragt. Sie wird mit der Prozedur „Bitwandel“ in eine 16-Bit-Binärzahl plus einer 8-Bit-Prüfsumme umgewandelt und sodann ausgedruckt. Nach Eingabe der Zahl kann noch zwischen normalem Ausdruck und schmalem Druck gewählt werden. Zum Ausdrucken wird mit ESC A der Zeilenvorschub verringert. Die Balken werden dreimal untereinander gedruckt, damit der Leser frei geführt werden kann. Mit

```
PROGRAM Barcode;
```

```
{*****
 *
 * K.-H. Rollke, Unna, Programm zum Erzeugen und Lesen *
 * von Barcodes fuer 16-Bit-Zahlen mit einer 8-Bit- *
 * Pruefsumme (Quersumme der Zahl).Das Programm *
 * benoetigt einen grafikfaehigen EPSON-Drucker. *
 * Es ist getestet mit EPSON-FX-80 und MX-80. *
 * *****}
```

```
USES APPLESTUFF;
```

```
VAR Wahl,Flag:CHAR;
    Drucker:INTERACTIVE;
    Bit:ARRAY[1..24] OF 0..1;
    Ref,Bar,Pruef,Nummer:INTEGER;
    Mode:BOOLEAN;
```

```
{*****}
```

```
PROCEDURE Bitwandel(VAR Zahl:INTEGER);
```

```
    { wandelt ganze Zahlen in Dualzahlen }
```

```
VAR Z,I:INTEGER;
    Bin:INTEGER;
```

Bild 1. Grundlage für konkrete Anwendungen: Das Pascal-Programm zum Erstellen und Lesen von Strichcodeausdrucken

```
BEGIN
```

```
    Bin:=128;
    Z:=Zahl DIV 256;
    FOR I:=1 TO 8 DO { niederw. Byte }
    BEGIN
        Bit[I]:=Z DIV Bin;
        Z:=Z MOD Bin;
        Bin:=Bin DIV 2;
```

```
    END;
    Z:=Zahl MOD 256;
    Bin:=128;
    FOR I:=9 TO 16 DO { hoeherw. Byte }
    BEGIN
        Bit[I]:=Z DIV Bin;
        Z:=Z MOD Bin;
        Bin:=Bin DIV 2;
```

```
    END;
    Pruef:=0; { Berechnung der Pruefsumme }
    Bin:=10000;
    Z:=Zahl;
    FOR I:=1 TO 5 DO
    BEGIN
        Pruef:=Pruef+(Z DIV Bin);
        Z:=Z MOD Bin;
        Bin:=Bin DIV 10;
```

```
    END;
    Bin:=128;
    Z:=Pruef;
    FOR I:=17 TO 24 DO
    BEGIN
        Bit[I]:=Z DIV Bin;
        Z:=Z MOD Bin;
        Bin:=Bin DIV 2;
```



```

END;
END; { of Bitwandel }

(*****)

PROCEDURE Drucke;

    { Ausdruck des Barcodes }

VAR N,I,J,Zaehler:INTEGER;
    Eins,Leer,esc:CHAR;
    B,Breite:CHAR;

BEGIN
    esc:=CHR(27); { Escape }
    Eins:=CHR(255); { Bitmuster senkr. Strich }
    Leer:=CHR(0); { Bitmuster Leer }

    PAGE(OUTPUT);
    WRITELN('Welche Zahl <0..32767> soll');
    WRITELN('gedruckt werden?');
    WRITELN;
    WRITE('Zahl: ');
    READLN(Nummer);
    WRITELN;
    WRITE('Normal breit oder S)chmal und fett?');
    READ(Breite);
    WRITELN(Breite);
    IF Breite IN ['S','s'] THEN B:='L' ELSE B:='K';

    IF (Nummer>32767) OR (Nummer<0) THEN
        BEGIN WRITELN('Zahl ZU GROSS, D.H. <0>!');
            Nummer:=0;
        END;
    Bitwandel(Nummer);

    REWRITE(Drucker,'PRINTER:');
    WRITE(Drucker,esc,'A',CHR(7),CHR(13),CHR(10));
    { stellt kleinen Zeilenvorschub ein }

    FOR I:=1 TO 3 DO
    BEGIN
        WRITE(Drucker,esc,B,CHR(127),CHR(1));
        { B="K" NORMAL DENSITY / B="L" DUAL DENSITY Mode }
        FOR N:=1 TO 30 DO WRITE(Drucker,Leer);
        Zaehler:=30;
        FOR J:=1 TO 2 DO
            WRITE(Drucker,Eins,Eins,Eins,Eins,Eins,Eins,Leer,Leer,
                Leer,Leer);
        FOR J:=1 TO 24 DO
        BEGIN
            IF Bit[J]=1 THEN
                BEGIN WRITE(Drucker,Eins,Eins,Eins,Eins,Eins,Eins);
                    Zaehler:=Zaehler+6 END
            ELSE
                BEGIN WRITE(Drucker,Eins,Eins);Zaehler:=Zaehler+2 END;
            WRITE(Drucker,Leer,Leer,Leer,Leer);
            Zaehler:=Zaehler+4;
        END;

        { Die Zeilenbreite von 384 Punkten muss aufgefullt
          werden: }
        FOR J:=1 TO 364-Zaehler DO WRITE(Drucker,Leer);

        WRITE(Drucker,CHR(0),CHR(13)); { NORMAL Mode }
    END;

    WRITELN(Drucker);
    WRITE(Drucker,esc,2); { normaler Zeilenvorschub }
    WRITE(Drucker,"      Zahl: ",Nummer);
    WRITELN(Drucker," - Pruef: ",Pruef);
    WRITELN(Drucker);
    CLOSE(Drucker);
END; { of Drucke }

(*****)

PROCEDURE Test;

    { fragt Paddles nach Status ab }

```

```

BEGIN
    Bar:=0;
    WHILE BUTTON(0)=Mode DO Bar:=Bar+1;
    Bar:=0;Mode:=(NOT Mode);
    WHILE BUTTON(0)=Mode DO Bar:=Bar+1;
    Mode:=(NOT Mode);
END; { OF Test }

(*****)

PROCEDURE Lese;

    { Lesen eines fertigen Barcodes mit Pruefsumme }

VAR Sum,Bin,Zahl,I:INTEGER;

BEGIN
    PAGE(OUTPUT);
    WHILE BUTTON(0)=TRUE DO BEGIN END;
    Mode:=FALSE;
    FOR I:=1 TO 2 DO Test;
    Ref:=Bar DIV 2;
    FOR I:=1 TO 24 DO
    BEGIN
        Test;
        IF Bar<=Ref THEN
            BEGIN Bit[I]:=0;Ref:=Bar+Bar DIV 2 END
        ELSE
            BEGIN Bit[I]:=1;Ref:=Bar DIV 2 END;
    END;
    NOTE(5,50); { Ton }

    Bin:=1;Zahl:=0;
    IF Bit[1]=1 THEN BEGIN WRITE(CHR(7),CHR(7),CHR(7),
        Fehler!<Bit 1>');
        EXIT(Lese) END;

    FOR I:=16 DOWNT0 1 DO
    BEGIN
        Zahl:=Zahl+Bin*Bit[I];
        Bin:=Bin*2;
    END;
    Bin:=1;Sum:=0;
    FOR I:=24 DOWNT0 17 DO
    BEGIN
        Sum:=Sum+Bin*Bit[I];
        Bin:=Bin*2;
    END;

    Bitwandel(Zahl);
    IF Pruef<>Sum THEN BEGIN
        WRITELN(CHR(7),CHR(7),CHR(7),
            Fehler!<Pruefsumme>');
        EXIT(Lese)
    END;

    WRITELN;WRITELN;
    FOR I:=1 TO 24 DO WRITE(Bit[I]);
    WRITELN;WRITELN;
    WRITELN('Gelesene Zahl: ',Zahl);
    WRITELN('Pruefsumme : ',Sum);
    END;{ of Lese }

(*****)

{ HAUPTPROGRAMM }

BEGIN
    REPEAT
        PAGE(OUTPUT);
        WRITELN('Bar - Code - Programm');
        WRITELN;WRITELN;
        WRITE('Drucken , Lesen , Ende?');
        READ(Wahl);
        CASE Wahl OF
            'L','l':BEGIN Lese;WRITELN;WRITELN('TASTE...');
                READ(Flag) END; 'D','d':Drucke;
        END
    UNTIL NOT (Wahl IN ['L','l','D','d']);
    PAGE(OUTPUT);
END.

```

ESC K... bei normalem Ausdruck (Normal Density Mode) oder ESC L... bei schmalen Druck (Dual Density Mode) wird die maximale Breite der zu druckenden Grafik vorgegeben (daher muß der Rest der Zeile mit Leerzeichen aufgefüllt werden. Die Grafikzeichen des Druckers lassen sich erzeugen durch CHR\$(x), wobei x die dezimale Darstellung einer Dualzahl ist, die das Bitmuster des Zeichens festlegt. Mit CHR\$(0) wird keine Nadel angesprochen, mit CHR\$(1) die untere Nadel, mit CHR\$(2) die zweite Nadel, mit CHR\$(3) die beiden unteren Nadeln usw. bis CHR\$(255) (d. h. alle 8 Nadeln werden angesprochen).

Beim Einlesen werden Änderungen des Logikpegels am Ausgang des Strichcodelesers erfaßt und die Längen der schwarzen Striche gezählt. Dabei ist REF die Referenzlänge, mit der die Breiten der Striche verglichen werden [1]. Beim Lesen der Striche wird die Referenzlänge laufend aktualisiert. Fehler beim Einlesen werden erkannt, indem die gelesene Zahl (16 Bit) in eine Dezimalzahl gewandelt und ihre Quersumme mit der Prüfsumme (8 Bit) verglichen wird. Die Fehlermeldung wird akustisch unterstützt durch dreimaliges Piepsen. Das Ende des Lesevorganges wird durch ein Brummen (NOTE (5,50)) gemeldet. Liest man Strichcodes ein, die mit normaler Breite gedruckt sind, so ergeben sich kaum Schwierigkeiten (sofern man natürlich die Richtung beibehält und alle Striche erfaßt). Bei Codes mit schmalen Strichen können aufgrund der

doch noch recht langsamen Leseschleife Fehler auftreten, wenn man den Leser nicht langsam genug bewegt.

Grundlage für konkrete Anwendung

Das beschriebene Programm würde sich schon als Bestandteil eines Bibliotheksprogrammes eignen. Sollen nur Teile (einzelne Prozeduren) verwendet werden, so ist darauf zu achten, daß einige Variablen für das ganze Programm definiert sind. Denkbar ist z. B. ein Programm, das Bücher (oder andere Artikel wie Schallplatten, Disketten etc.) mit laufenden oder systematischen Nummern versieht (indem entsprechende Etiketten gedruckt werden) und diese Nummern mitsamt Angaben über die so erfaßten Bücher speichert. Außerdem könnten Benutzerausweise gleichermaßen gedruckt werden. Bei der Ausleihe von Büchern werden die Benutzernum-

mer und die Buchnummer eingelesen, und so wird die Ausleihe des Buches im Rechner vermerkt. Sollten größere Zahlen als die vom Programm zugelassenen benötigt werden, muß man die Bitwandel-Prozedur überarbeiten. Außerdem empfiehlt es sich insbesondere bei größeren Zahlen, den Leseteil als Assembleroutine zu schreiben, da die Lesesicherheit dadurch erheblich verbessert würde und man aufgrund schmalerer Striche den Platzbedarf reduzieren könnte.

Dieses Programm soll eine Anregung darstellen – weitere Verfeinerungen müssen sich am konkreten Problem orientieren.

Literatur

- [1] Hofer, Rudolf: Strichcode drucken und lesen. mc 1983, Heft 4, Seite 66 und 91.
- [2] Lesestift für mc-Programme. mc 1981, Heft 1, Seite 45

Apple-Grafik füllt eine DIN-A4-Seite

Zu unserem Beitrag auf Seite 11: Das Nachfolgemodell des Epson MX-82, der Epson FX-80, bietet ebenfalls einen Druckmodus mit 576 Punkten/Zeile. Die Grafik wird allerdings etwas anders initialisiert, außerdem ist der Zeilenvorschub nicht identisch. Da der Rest des

Programmes unverändert blieb, ist hier nur der Hex-Dump wiedergegeben. Beim aufrufenden Basic-Programm mußte die Option doppelte Druckdichte entfallen, da der FX-80 maximal 960 Punkte/Zeile ansprechen kann.

Wolfgang Ebner



Zahl:1 - Pruef:1



Zahl:2 - Pruef:2



Zahl:127 - Pruef:10



Zahl:255 - Pruef:12



Zahl:32767 - Pruef:25



Zahl:528 - Pruef:15

Bild 2. Einmal breit, einmal schmal: Die Lesesicherheit ist größer, wenn das Pascal-Programm mehr Zeit zum Zählen der Strichbreite hat. Wer Platzprobleme hat, sollte den Leseteil durch eine Assembleroutine ersetzen [1]

Hex-Dump und Basic-Rahmenprogramm zur Ausgabe einer Apple-Grafik auf den Drucker FX-80

```
0300: A9 20 85 E6 A9 1B 20 82 03 A9 33 20 82 03 A9 11
0310: 20 82 03 20 C4 03 A9 01 85 E1 A9 15 85 E0 A9 00
0320: 85 E2 20 BB 03 A9 00 85 01 85 00 E6 E0 E6 E0 A9
0330: 01 C5 E0 D0 02 E6 E1 06 00 06 00 06 00 20 A5 03
0340: 05 00 85 00 E6 01 20 B9 03 A5 01 C9 02 D0 EB A2
0350: 03 A5 00 EA EA 20 82 03 CA D0 FA E6 E2 A5 E2 C9
0360: C0 D0 C2 C6 E0 20 B9 03 A9 FF C5 E1 D0 E0 A9 FD
0370: C5 E0 D0 AA 20 C4 03 A9 1B 20 82 03 A9 32 20 82
0380: 03 60 2C C1 C1 30 FB BD 90 C0 60 A9 1B 20 82 03
0390: A9 2A 20 82 03 A9 05 20 82 03 A9 40 20 82 03 A9
03A0: 02 20 82 03 60 A6 E0 A4 E1 A5 E2 20 11 F4 A5 30
03B0: 29 7F 31 26 F0 02 A9 07 60 C6 E0 A9 FF C5 E0 D0
03C0: 02 C6 E1 60 A9 0D 20 82 03 A9 0A 20 82 03 60
```

```
100 REM HGR DUMP für Epson FX80; W.Ebner 4/83
105 REM *****
110 PRINT CHR$(4) "BLOAD FX-DUMP.BIN"
120 TEXT : HOME
130 PRINT TAB(10) "H G R - D U M P"
140 PRINT TAB(10) "*****"
150 PRINT : PRINT
160 PRINT "Optionen : "
170 PRINT : PRINT
200 PRINT "1 - Invertieren"
210 PRINT
220 PRINT "2 - Bildschirmseite 2"
230 PRINT : PRINT
240 PRINT "9 - Programmende"
250 VTAB 23
260 PRINT SPC(30);
270 HTAB 1
280 INPUT "Eingabe: "; Z$
290 Z = VAL(Z$)
300 IF Z < 3 AND Z > 0 THEN 350
310 IF Z > 9 THEN 250
320 HOME : PRINT "HGR DUMP geladen. Programmstart mit '&'."
330 POKE 1014,0: POKE 1015,3: REM '&' anschließen
340 PRINT : END
350 IF Z = 1 THEN POKE 851,73: POKE 852,63
360 IF Z = 2 THEN POKE 769,64
380 GOTO 250
```


Jochen Schafft, Volker Weilacher

ROM-Routinen des Applesoft-Basic-Interpreters

Die ROM-Routinen von PET, CBM und TRS-80 wurden schon in früheren Ausgaben der mc und im Franzis-Sonderheft „Mikrocomputer-Anwendungen“ zusammengestellt [1, 2]. Im folgenden werden in ähnlicher Weise alle wichtigen Adressen des Applesoft-Basic-Interpreters genannt.

Da der Basic-Interpreter des Apple II plus ebenso wie die Interpreter der CBM-Serie aus dem Hause Microsoft stammt, liegt es nahe, mit Hilfe eines ROM-Listings CBM-Programme auf den Apple zu übertragen. Dies ist relativ einfach möglich, da sich der größte Teil der Interpreter-Routinen in der Funktionsweise nicht unterscheidet. Lediglich bei den Ein-/Ausgabeoperationen sind größere Eingriffe nötig (Tastatur, Bildschirm, Diskettenstation, Kassette usw.).

Noch zwei weitere Gruppen von Apple-Anwendern haben großes Interesse an einer Aufstellung aller wichtigen Adressen des Basic-Interpreters:

Zum ersten all jene, die schon immer einmal ergründen wollten, wie ein Interpreter einer höheren Sprache arbeitet. Eventuell gab ein Fehler des Microsoft-Basic den Anstoß, dessen Geheimnisse zu lösen. Besitzer einer 16-K-RAM-Karte oder des „Language Systems“ können sogar selbst den mitgelieferten Interpreter abändern und mit ihrer eigenen (hoffentlich) fehlerfreien Version arbeiten.

Die zweite Gruppe besteht aus all jenen, denen der Befehlssatz des Apple nicht ausreicht und die ihn durch „hausgemachte“ Befehle ergänzen wollen. Als Anleitung hierfür dienen die bekannten

Basic-Erweiterungen des CBM (Toolkit, sm-Kit etc.). Solche Ergänzungen lassen sich beim Apple prinzipiell auf drei verschiedenen Wegen durchführen:

1. Bei der Ausführung eines Programms als auch direkt eingegebener Befehle werden alle Befehlscodes über eine Routine in der Zero-Page geholt (CHRGET-Routine). An dieser Stelle kann leicht ein Sprung in die eigene Befehlscodier-Routine eingefügt werden. Der Nachteil dieses Verfahrens ist jedoch eine verminderte Verarbeitungsgeschwindigkeit.
2. Jede Ein- und Ausgabe erfolgt beim Apple über zwei Vektoren in der Zero-Page (näheres dazu im „Reference Manual“). Befehle, die nur im Direkt-Modus verfügbar sein sollen, können dadurch implementiert werden, daß man den Eingabevektor auf die eigene Routine umlenkt. Diese Methode hat den Vorteil, daß die Verarbeitungsgeschwindigkeit bei der Abarbeitung von Programmen nicht beeinflusst wird. Probleme ergeben sich jedoch bei gleichzeitiger Verwendung eines Diskettenlaufwerks, da das Diskettenbetriebssystem des Apple auf die gleiche Weise in den Basic-Interpreter eingreift. Das DOS-Handbuch be-

schreibt, wie man diese Probleme umgehen kann. Eine Erkennung der neuen Befehle innerhalb von Programmen kann man erreichen, indem man auch den Ausgabevektor „umbiegt“ und alle Befehle nur innerhalb eines PRINT-Ausdruckes zuläßt. Ein spezielles Control-Zeichen, ähnlich dem Control-D beim DOS, vereinfacht die Befehlserkennung.

3. Das Microsoft-Basic des Apple besitzt einen eigenen Befehl zur Erweiterung des Befehlssatzes: das „&“-Zeichen. Sobald der Interpreter auf dieses Zeichen stößt, springt er nach 3F5. An dieser Stelle kann man einen JMP-Befehl in die eigene Maschinenroutine ablegen. Nach dem Basic-Kaltstart befindet sich in 3F5 bis 3F7 ein Sprungbefehl nach FF58 (auf einen RTS-Befehl). Der Vorteil dieses Verfahrens ist, daß die Verarbeitungsgeschwindigkeit nicht beeinflusst wird und der &-Befehl sowohl im Direkt- als auch im Programmmodus erkannt wird. Die einzige Bedingung an die neuen Befehle ist, daß sie bzw. die Befehlszeile in der sie stehen mit einem „&“ beginnen. Diese Technik wird auch in dem Renumber-Programm auf der System-Master-Diskette benutzt.

Benutzung des Stack beim Apple

Der Stack (0100...01FF) wird nicht nur zur Speicherung der Rücksprungadressen der Interpreter-Routinen und der eigenen Maschinenspracheprogramme benutzt. Auf ihm legt der Interpreter auch die Parameter einer FOR-NEXT-Schleife und des GOSUB-Befehls ab. Des weiteren werden bei der Auswertung arithmetischer Ausdrücke die Zwischenergebnisse bei offenen Operationen auf den Stack „geschoben“.

Literatur

- [1] Martin, Reinhold; Smode, Dieter: ROM und RAM bei PET und CBM. Mikrocomputer-Anwendungen (Sonderheft des Franzis-Verlags), Seite 58...69.
- [2] Röckrath, Luidger: Der geknackte TRS-80. mc 1981, Heft 1, S. 46...48 und Heft 2, S. 37...44.
- [3] Handle, Franz: Zahlendarstellung im PET. Hobbycomputer 2 (Sonderheft des Franzis-Verlags), S. 19...22.

RAM-Bereich (Zero-Page):

00-02	JMP-Befehl für Basic-Warmstart	7B-7C	DATA-Zeilenummer
03-05	JMP-Befehl: Sprung nach DB3A (String ab A/Y drucken)	7D-7E	DATA-Zeiger
06-09	unbenutzt	7F-80	Zeiger für Eingabeoperationen
0A-0C	JMP-Befehl für USR-Befehl. Nach Basic-Initialisierung: JMP E199 (Ausgabe von „ILLEGAL QUANTITY ERROR“)	81-82	letzter Variablenname
0D-0E	Hilfsregister für Vergleiche von einzelnen Zeichen (z. B. in der Druckroutine für Strings)	83-84	Zeiger auf zuletzt benutzte Variable
0F	Anzahl der Dimensionen des letzten Feldes	85-86	Zeiger auf laufende Variable (bei FOR und LET-Anweisung)
10	DIM-Flag	87-88	Zwischenspeicher für CHRGET-Zeiger bei Eingabeoperationen
11	Flag für Stringvariable (bei Stringvariable = FF)	89	Kennzeichen für die einzelnen Vergleichsoperationen
12	Flag für Integervariable (bei Integer = 80, sonst 00)	8A-8B	Zeiger für DEF FN
13	Flag für GARBAGE COLLECTION: negativ, wenn bereits durchgeführt; Zähler bei Codierung der Basic-Befehle	8B	Flag für GARBAGE COLLECTION
14	Flag für Variablenbehandlung: 00: Normale Behandlung aller Variablenarten 80: Keine Stringvariablen erlaubt; bei Erkennen einer Stringvariablen „SYNTAX ERROR“ ausgeben! 40: Es handelt sich um eine Feldvariable, obwohl dem Variablennamen keine Klammer folgt, z. B. bei STORE	8C-8D	Zeiger für Stringverarbeitung; zeigt auf gültigen Stringdeskriptor
15	Flag zur Unterscheidung der Eingabeoperationen: 00: INPUT 40: GET 98: READ	8E	unbenutzt
16	Flag dafür, daß bei der SIN-Berechnung das Vorzeichen des Argumentes geändert wurde (Wert lag im 3. bzw. 4. Quadranten); wird in TAN-Routine benötigt	8F	Länge einer Variablen im Speicher
17-19	unbenutzt	90-92	JMP-Befehl für Funktionsaufruf
1A-1B	Zeiger für HGR-Routinen	93-97	FAC 3 (5 Bytes); FAC = Fließkommaakkumulator
1C	Farbmaske, eventuell mit 7F Exklusiv-ODER-verknüpft	94-95	Zeiger auf das erste Datenbyte einer Feldvariablen bei Speicherblockverschiebung: Endadresse + 1
1D	Zähler für HGR-Routinen	96-97	FAC 4 (5 Bytes)
1E-1F	unbenutzt	98-9C	Zeiger für Speicherblockverschiebung, Suchzeiger, Zeiger auf Variablenkopf
20-4F	belegt durch System-Monitor	9B-9C	FAC 1 (5 Bytes): Exponent und Mantisse, jedoch nicht in Normalform; Mantisse ohne Vorzeichen
24	horizontale Cursorposition	A2	Vorzeichen für FAC 1
25	vertikale Cursorposition	A3	Zähler für Polynomrechnung
26-27	Zeiger für HGR-Routinen: enthält RAM-Adresse des zuletzt angesprochenen Punktes	A4	allg. Register für Arithmetik
2C-2D	Koordinaten für Blockgrafik	A5-A9	FAC 2 (5 Bytes)
30	Maske für Setzen bzw. Löschen eines Bildschirmpunktes in HGR	AA	Vorzeichen für FAC 2
3C-3D	Anfangsadresse für Kassettenroutinen	AB	Vorzeichen FAC 1 multipliziert mit Vorzeichen FAC 2 (Exklusiv-ODER-Verknüpfung der Vorzeichen)
3E-3F	Endadresse für Kassettenroutinen	AC	Schutzstelle für FAC 1
50-51	Register für Integer-Zahlen	AD	Länge eines Feldelementes (2, 3 oder 5 Bytes)
52	Zeiger für Deskriptorenstack	AD-AE	bei DIM: Platzbedarf für Daten im Feld; bei Feldvariablenbehandlung: Berechnung der Position eines Feldelementes
53-54	Adresse des zuletzt verwendeten Strings	AF-B0	Ende des Basic-Textes, wird durch LOMEM-Befehl nicht beeinflusst
55-5D	Deskriptorenstack für Stringverarbeitung (je 3 Bytes)	B1-C8	CHRGET-Routine: holt ein Zeichen aus Basic-Text, wird auch bei Eingabeoperationen benutzt (Input, Read)
5E-5F	Register für ind. Sprung bei Auswertung arithmetischer Ausdrücke	B8-B9	CHRGET-Zeiger, Zeiger auf letztes Basic-Zeichen (bzw. Befehl)
60-61	Zeiger, allg. Verwendung (Verschieberoutine, VAL-Routine)	C9-CD	RND-Register: enthält letzte Zufallszahl in FP-Darstellung
62-65	FP-Hilfsregister für Multiplikation und Division (nur Mantisse)	CE-CF	unbenutzt
64-65	Hilfsregister für Behandlung von Feldern	D0-D2	Hilfsregister für SHAPE-Verarbeitung
66	nicht benutzt	D3	enthält ROT-Wert mod 16: Winkel für DRAW bezogen auf nächste rechtwinklige Achse
67-68	Beginn des Basic-Textes; wird bei Initialisierung des Interpreters auf 0801 gesetzt	D4-D5	Register für HGR-Routinen
69-6A	Beginn der Variablen, LOMEM-Wert	D6	Flag für Autostart (neg., wenn Autostart)
6B-6C	Beginn der Feldvariablen, Ende der Variablen + 1	D7	unbenutzt
6D-6E	Ende der Feldvariablen	D8	Flag für ONERR: negativ nach Aktivierung von ONERR
6F-70	unteres Ende des Stringspeichers, bewegt sich nach unten	D9	unbenutzt
71-72	allg. Zeiger	DA-DB	enthält Zeilenummer nach Error-Zustand (für RESUME)
73-74	HIMEM: höchste verfügbare Speicherzelle + 1	DC-DD	enthält RAM-Adresse des letzten Befehls vor dem Error-Zustand
75-76	Zeilenummer der Basic-Zeile, die gerade ausgeführt wird	DE	enthält Index des letzten aufgetretenen Fehlers (Tabelle in Applesoft-Handbuch, Seite 136)
77-78	Zeilenummer der Basic-Zeile, bei der Programmausführung unterbrochen wurde	DF	Zwischenspeicher für Stackpointer während der Ausführung der Basic-ONERR-Routine; RESUME setzt Stackpointer wieder auf alten Wert
79-7A	Zeiger auf zuletzt ausgeführten Befehl	E0-E1	X-Koordinate für HPLOT, DRAW
		E2	Y-Koordinate für HPLOT, DRAW
		E3	unbenutzt
		E4	Farbmaske
		E5	Nummer der Spalte, in der sich das zuletzt angesprochene Bit einer HGR-Seite befindet
		E6	Flag zur Unterscheidung der HGR-Seiten (20 für erste Seite, 40 für zweite Seite)
		E7	SCALE

Das Apple-Sonderheft

E8-E9	Zeiger auf SHAPE TABLE
EA	Hilfsregister für HGR-Routinen
EB-EF	unbenutzt
F0	Hilfsregister für Blockgrafik
F1	SPEED-Wert, Zweierkomplement des Speed-Wertes für Verzögerungsschleife
F2	Flag für TRACE: negativ, wenn TRACE aktiviert
F3	FLAG für INVERSE und FLASH: 3F = INVERSE, FF = NORMAL, 7F = FLASH
F4-F5	RAM-Adresse des momentan gültigen ONERR-Statements
F6-F7	Zeilennummer der Basic-Zeile in der momentan gültiges ONERR-Statement steht
F8	Zwischenspeicher für Stackpointer
F9	ROT-Wert
FA-FF	unbenutzt

ROM-Bereich (D000-F7FF)

D000-D07F	Sprungtabelle für Basic-Befehle. Enthält Anfangsadresse - 1 (Aufruf für RTS) D000: END, FOR, NEXT, DATA D008: INPUT, DEL, DIM, READ D010: GR, TEXT, PR#, IN# D018: CALL, PLOT, HLIN, VLIN D020: HGR2, HGR, HCOLOR, HPLLOT D028: DRAW, XDRAW, HTAB, HOME D030: ROT=, SCALE=, SHLOAD, TRACE D038: NOTRACE, NORMAL, INVERSE, FLASH D040: COLOR=, POP, VTAB, HIMEM: D048: LOHMEM:, ONERR, RESUME, RECALL D050: STORE, SPEED=, LET, GOTO, D058: RUN, IF, RESTORE, & D060: GOSUB, RETURN, REM, STOP D068: ON, WAIT, LOAD, SAVE D070: DEF, POKE, PRINT, CONT D078: LIST, CLR, GET, NEW
D080-D0B1	Sprungtabelle für arithmetische Operationen (Teil 1) Enthält Anfangsadressen der einzelnen Routinen D080: SGN, INT, ABS, USR, FRE, SCRN, PDL, POS D090: SQR, RND, LOG, EXP, COS, SIN, TAN, ATN D0A0: PEEK, LEN, STR\$, VAL, ASC, CHR\$, LEFT\$, RIGHT\$ D0B0: MID\$
D0B2-D0CF	Sprungtabelle für arithmetische Operationen (Teil 2) Jeder Operation sind drei Bytes zugeordnet: Das erste Byte dient zur Erkennung des „Vorrangs“ einzelner Operationen. Die beiden folgenden ergeben die Anfangsadresse der Routine D0B2: Addition: 79, E7C0 D0B5: Subtraktion: 79, E7A9 D0B8: Multiplikation: 7B, E981 D0BB: Division: 7B, EA68 D0BE: Potenzierung: 7D, EE96 D0C1: AND: 50, DF54 D0C4: OR : 46, DF4E D0C7: Vorzeichenwechsel: 7F, EECF D0CA: NOT: 7F, DE97
D0D0-D25F	Tabelle der Befehlsörter in ASCII. Beim letzten Zeichen eines jeden Befehls ist Bit 7 gesetzt
D260-D364	Systemmeldungen in ASCII. Jede Systemmeldung endet mit dem Byte 00. Auswahl einer Fehlermeldung mit Hilfe des X-Registers D260: „NEXT WITHOUT FOR“ (X=00) D270: „SYNTAX“ (X=10) D276: „RETURN WITHOUT GOSUB“ (X=16) D28A: „OUT OF DATA“ (X=2A) D295: „ILLEGAL QUANTITY“ (X=35) D2A5: „OVERFLOW“ (X=45) D2AD: „OUT OF MEMORY“ (X=4D)

D2BA: „UNDEF'D STATEMENT“ (X=5A)	
D2CB: „BAD SUBSCRIPT“ (X=6B)	
D2D8: „REDIM'D ARRAY“ (X=78)	
D2E5: „DIVISION BY ZERO“ (X=85)	
D2F5: „ILLEGAL DIREKT“ (X=95)	
D303: „TYPE MISMATCH“ (X=A3)	
D310: „STRING TOO LONG“ (X=B0)	
D31F: „FORMULA TOO COMPLEX“ (X=BF)	
D332: „CAN'T CONTINUE“ (X=D2)	
D340: „UNDEF'D FUNKTION“ (X=E0)	
D350: „ERROR“ + Bell	
D358: „IN“	
D35D: CR + „BREAK“ + Bell	
D365-D392	Stack nach Parameter einer offenen FOR-NEXT-Schleife absuchen. Wenn Parameter der gesuchten Schleife nicht vorhanden, dann Z=1
D393-D3D5	Platz für neue Basic-Zeile schaffen
D39A-D3D5	Speicherblock verschieben. Anfangsadresse des Blocks in 9B/9C, Endadresse + 1 in 96/97, Zieladresse + 1 in 94/95
D3D6-D3E2	Droht Stacküberlauf? Wenn ja, dann Fehlermeldung
D3E3-D40F	Reicht der verfügbare Speicherplatz noch aus? Wenn nicht, dann GARBAGE COLLECTION durchführen; wenn nötig, „OUT OF MEMORY ERROR“ ausgeben
D410-D43B	„OUT OF MEMORY ERROR“ ausgeben
D412-D43B	Fehlermeldung ausgeben. Art des Fehlers wird durch X-Register bestimmt Falls ONERR-Flag gesetzt, dann Sprung nach F2E9
D42D-D43B	„ERROR“ + Bell ausgeben
D434-D43B	Wenn HByte der Zeilennummer ≠ FF (also Programm-Modus), dann „IN“ mit Zeilennummer ausgeben
D43C-D459	Basic-Warmstart. Prompt-Meldung und neue Zeile holen; CHRGET-Zeiger auf 01FF setzen Ist erstes Zeichen eine Ziffer? Wenn ja, dann weiter bei D45C; andernfalls Sprung nach D805
D45C-D52B	neue Zeile in Basic-Text übernehmen
D45C-D4B9	Alte Basic-Zeile mit gleicher Nummer löschen; wurde nur Zeilennummer eingegeben, so erfolgt ein Sprung nach D4F2; andernfalls weiter...
D4BA-D4F4	Neue Zeile in Basic-Text einfügen. CHRGET-Zeiger auf Programmstart setzen
D4F5-D52B	Links neu berechnen. Sprung nach D43C (Basic-Warmstart)
D52C-D552	Eine Zeile von der Tastatur holen. Setzt Prompt-Zeichen = 80 und ruft dann FD6A auf. Löscht Bit 7 aller Zeichen im Input-Puffer und fügt das Byte 00 an
D553-D558	holt ein Zeichen von der Tastatur und löscht Bit 7
D559-D619	Codierung der Basic-Befehle. Alle Blanks zwischen Befehlswörtern werden entfernt. Falls das Auto-Start-Flag gesetzt ist, erfolgt ein Zwangs-RUN
D61A-D648	Sucht nach der Basic-Zeile, deren Nummer in 50/51 steht. Ist die gewünschte Zeile vorhanden, so ist C=1 und ihre Adresse in 9B/9C. Andernfalls ist C=0, und die Adresse der nächsten Basic-Zeile befindet sich in 9B/9C
D649-D696	NEW ausführen
D66A-D696	CLEAR ausführen
D683-D696	Stack initialisieren, CONT unmöglich machen
D697-D6A4	CHRGET-Zeiger auf Programmstart setzen (Inhalt von 67/68-1)
D6A5-D765	LIST-Routine
D766-D7D1	FOR-Routine. Schiebt Parameter der FOR-NEXT-Schleife auf den Stack
D7D2-D827	Befehlsausführung. Überprüfung ob Control-C gedrückt. Wenn ja, dann „BREAK IN“... Überprüft, ob CHRGET-Zeiger auf ein Byte 00 zeigt (Ende einer Zeile). Wenn ja, dann wird nächste Zeilennummer aus Basic-Text geholt und in 75/76 abgespeichert. Bei gesetztem TRACE-Flag wird die neue Zeilennummer zusätzlich noch auf dem Bildschirm ausgegeben
D828-D848	Aufruf der Basic-Routinen. Die Adresse wird auf den Stack geschoben. RTS ruft die Routine auf

- D849–D857 RESTORE-Routine. Der DATA-Zeiger 7D/7E wird auf den Programmanfang ((67/68)–1) gesetzt
- D858–D85F Überprüft ob Control-C gedrückt ist. Wenn ja, dann weiter bei D860. Andernfalls RTS
- D860–D86D Holt laufendes Zeichen von Tastatur. Bei gesetztem ONERR-Flag erfolgt Sprung nach F2F9. Sonst weiter mit...
- D86E–D895 STOP ausführen
- D870–D895 END ausführen. Unterscheidung zwischen STOP und END mit Hilfe des C-Flags: C=0 \triangle END; C=1 \triangle STOP. Befindet sich der Rechner im Programm-Modus, werden der CHRGET-Zeiger nach 79/7A und die Zeilennummer nach 77/78 übertragen
- D88A–D895 Bei gesetztem Carry-Flag „BREAK IN“... ausgeben
- D896–D8AF CONT-Routine
- D8B0–D8C8 SAVE-Routine
- D8C9–D8EF LOAD-Routine
- D8F0–D900 Unterprogramm für LOAD und SAVE. Setzt Anfangsadresse der Lese- und Schreibroutinen des Autostartmonitors auf 0050 und die Endadresse auf 0052
- D901–D911 setzt Start- und Endadresse des Programms für LOAD und SAVE
- D912–D920 RUN ausführen
- D921–D93D GOSUB-Routine
- D93E–D96A GOTO-Routine
- D96B–D978 RETURN- und POP-Routine Teil 1
- D979–D97D „RETURN WITHOUT GOSUB ERROR“ ausgeben
- D97C–D97D „UNDEF'D STATEMENT ERROR“ ausgeben
- D97E–D980 Sprung nach DEC9 („SYNTAX ERROR“ ausgeben)
- D981–D9A2 RETURN und POP, Teil 2
- D995–D9A2 DATA-Routine. Rückt CHRGET-Zeiger bis zum nächsten Befehl vor
- D9A3–D9C8 Nächsten Befehl suchen
- D9A6–D9C8 Nächste Zeile suchen
- D9C9–D9EB IF-Routine
- D9DC–D9EB REM-Routine: CHRGET-Zeiger auf nächste Zeile vorrücken
- D9EC–DA0B ON-Routine
- DA0C–DA45 Adresse (Festkommazahl zwischen 0 und 63999) aus Basic-Text holen. Ergebnis steht in 50/51. Ist die Adresse größer als 63999, dann „SYNTAX ERROR“ ausgeben
- DA46–DA79 LET-Hauptroutine: Bestimmt die links vom Gleichheitszeichen stehende Variable. Fehlt das Gleichheitszeichen, so erfolgt „SYNTAX ERROR“. Speichert Zeiger auf Variable in 85/86; schiebt Inhalt von 11 und 12 (Variablenart-Flags) auf Stack. Ruft dann DD7B auf
- DA7A–DACE Neuen String abspeichern (als Folge von LET)
- DACF–DAFA PRINT-Routine (Teil 1)
- DAFB–DB02 CR ausgeben
- DB03–DB56 PRINT-Routine (Teil 2)
- DB3A–DB56 Drückt String ab A/Y. Das Ende des Strings muß durch 00 oder 22 gekennzeichnet sein
- DB57–DB70 Blank ausgeben
- DB5A–DB70 „?“ ausgeben
- DB5C–DB70 ASCII-Zeichen, das im Akkumulator steht, ausgeben
- DB69–DB70 Verzögerung in Abhängigkeit vom Inhalt der Zelle F1 (SPEED-Wert) durchführen
- DB71–DB9F Fehlerbehandlung bei Eingabeoperationen (GET, INPUT und READ)
- DB7B–DB85 Zeilennummer der DATA-Zeile nach 75/76 übertragen. Sprung nach DEC9: „SYNTAX ERROR IN“...
- DB90–DB9F „?REENTER“ ausgeben
- DBA0–DBB1 GET-Routine. Benutzt einen Teil der READ-Routine
- DBB2–DBDB INPUT-Routine. Benutzt einen Teil der READ-Routine
- DBDC–DCD6 READ-Routine. Einsprung für READ: DBE2
- DCD7–DCDD „?EXTRA IGNORED“ ausgeben
- DCDF–DCEF Text: „?EXTRA IGNORED“ + CR
- DCEF–DCF8 Text: „?REENTER“ + CR
- DCF9–DDF3 NEXT-Routine
- DD54–DD66 Nach Fertigstellung der laufenden FOR-NEXT-Schleife: FOR-NEXT-Parameter aus Stack entfernen. Handhabt NEXT mit mehreren Laufvariablen
- DD67–DD75 Berechnet numerischen Ausdruck. Legt Ergebnis in FAC 1 ab. Wenn nötig, „TYPE MISMATCH ERROR“ ausgeben
- DD76–DD7A „TYPE MISMATCH ERROR“ ausgeben
- DD7B–DE5F Auswertung eines Terms (numerisch und String)
- DD7B–DD84 Auswertung eines Terms vorbereiten: CHRGET-Zeiger um eins erniedrigen und ein Byte 00 auf den Stack schieben (dient zur Erkennung der letzten offenen Operation)
- DD98–DDB3 Bei Vergleichsoperationen: Art des Vergleichs feststellen und Flag (89) entsprechend setzen: 01 für >, 02 für =, 03 für >=, 04 für <, 05 für \neq
- DE10–DE34 Schiebt Inhalt von FAC 1 auf Stack. Rücksprungadresse wird zuvor nach 5E/5F gerettet. Rücksprung erfolgt mit JMP indirekt
- DE35–DE5F Überträgt Argument vom Stack nach FAC 1 und ruft „gemerkte“ Routine (Anfangsadresse – 1 befindet sich auf Stack) auf. Ist das erste Byte auf dem Stack 00 (Kennzeichen dafür, daß keine Operation mehr offen ist), so wird die Auswertung des Terms abgebrochen
- DE60–DE97 Konstante bzw. Variable aus Basic-Text holen
- DE98–DEA3 NOT-Routine
- DEA4–DEC8 Auswertung eines Terms (2. Teil). Laufender Funktionscode im Akku. Für (Akku)=C2: Sprung nach DEAB (FN auswerten). Für (Akku)>=D2: Sprung nach DF0C (Basic-Funktion auswerten)
- DEB2–DEC8 Auswertung eines Terms in Klammern
- DEC9–DECD „SYNTAX ERROR“ ausgeben
- DECE–DED4 Hilfsroutine zur Auswertung eines Terms
- DED5–DEF8 Wert der folgenden Variable nach FAC 1 bringen. Existiert Variable noch nicht, wird sie neu definiert
- DEF9–DF0B SCRIN-Routine
- DF0C–DF4E Auswertung num. und String-Funktionen
- DF4F–DF54 OR-Verknüpfung durchführen
- DF55–DF64 AND-Verknüpfung durchführen
- DF65–DFCC Gemeinsame Routine für alle Vergleichsoperationen
- DF6A–DF7C Vergleich numerischer Größen
- DF7D–DFCC Vergleich zweier Strings
- DFCD–DFD5 PDL-Routine
- DFD6–DFE2 DIM-Routine
- DFE3–DFFF Variablenname aus Basic-Text holen und Variable suchen (Teil 1)
- E000–E002 JMP-Befehl nach F128: Initialisierung des Basic-Interpreters
- E003–E005 JMP-Befehl nach D43C: Basic-Warmstart
- E007–E07C Variablenname aus Basic-Text holen und Variable suchen (Teil 2)
- E053–E07C Numerische Variable suchen. Der Variablenname befindet sich dabei in 81/82. Gefunden \rightarrow E0DE. Falls Variable nicht vorhanden, Sprung nach E087
- E07D–E086 Überprüft, ob Zeichen im Akku ein Buchstabe ist. Wenn ja, dann C=1; andernfalls C=0
- E087–E099 Neue Variable einrichten
- E09A–E09B FP-Konstante mit dem Wert 0
- E09C–E0CC Feldvariablen um 7 Bytes nach oben verschieben
- E0CD–E0DD Neue Variable = 0 setzen
- E0DE–E0EC Speicherplatz der Variablen in 83/84 abspeichern (Inhalt von 9B/9C + 2)
- E0ED–E0FD Anfangsadresse der Daten im Feld bestimmen (nach Dimensionierungsbytes): Inhalt von 9B/9C + 5 + 2 \times Anzahl der Dimensionen nach 94/95
- E0FE–E101 FP-Konstante mit dem Wert –32768
- E102–E11D INT-Variable aus Basic-Text holen
- E11E–E195 Suchen eines Feldes. Falls gewünschtes Feld nicht vorhanden, Sprung nach E1B8
- E196–E19D „BAD SUBSCRIPT ERROR“ ausgeben
- E199–E19D „ILLEGAL QUANTITY ERROR“ ausgeben
- E19E–E1A9 Falls DIM-Flag (10) gesetzt: „REDIM'D ARRAY ERROR“
- E1AA–E1B7 Setzt 94/95 auf Beginn des Datenfeldes und überprüft die Dimensionierung. Wenn nötig, „BAD SUBSCRIPT ERROR“ ausgeben
- E1B8–E24A Neues Feld einrichten

- E24B-E2AC Gewünschtes Feldelement suchen. Berechnet Speicheradresse aus Anfangsadresse des Feldes, Anzahl der Dimensionen und den angegebenen Indizes
- E2AD-E2DD Unterprogramm für Feldbehandlung: multipliziert Inhalt von 64/65 mit Inhalt von AD/AE. Ergebnis in Y/X
- E2DE-E2FE FRE-Routine
- E2FF-E305 POS-Routine
- E306-E312 Befindet sich der Rechner im Direkt-Modus, dann „ILLEGAL DIRECT ERROR“ ausgeben
- E313-E340 DEF-Routine
- E341-E351 Syntax einer DEFFN-Zeile prüfen
- E354-E3A8 FN auswerten
- E3A9-E3C4 Holt alten CHRGET-Zeiger und FN-Variablenwert vom Stack
- E3C5-E3D4 STR\$-Routine: wandelt Inhalt von FAC 1 in String um
- E3D5-E3E6 Unterprogramm für Stringverarbeitung
- E3E7-E451 Länge eines Strings bestimmen: sucht nach dem Endzeichen 00 bzw. dem zweiten Anführungszeichen. Anfangsadresse des Strings in A/Y. Nimmt Deskriptor in Deskriptorentabelle auf und kopiert, wenn nötig, String in oberen RAM-Bereich
- E430-E434 „FORMULA TOO COMPLEX ERROR“ ausgeben
- E452-E473 Zeiger für unteres Ende des Stringspeichers korrigieren. Wird um die Länge des neuen Strings (in A) herabgesetzt. Reicht Speicherplatz für neuen String nicht mehr aus, wird die GARBAGE-COLLECTION-Routine angesprungen
- E474-E483 Überprüft, ob GARBAGE-COLLECTION bereits durchgeführt wurde. Falls ja, dann „OUT OF MEMORY ERROR“ ausgeben, andernfalls weiter mit...
- E484-E518 GARBAGE-COLLECTION durchführen: entfernt nicht mehr benötigte Strings im Speicher (Teil 1)
- E519-E561 Überprüft, ob Variable ab 5E/5F Stringvariable ist und ob String zwischen (9B/9C) und (6F/70) liegt. Falls ja, wird Anfangsadresse dieses Strings nach 9B/9C übertragen. Diese Routine dient zur Bestimmung der höchsten RAM-Adresse eines gültigen Strings (für GARBAGE-COLLECTION).
- E562-E596 GARBAGE-COLLECTION Teil 2: löscht nicht mehr benötigten String
- E597-E5D3 Verkettung zweier Strings durchführen
- E5D4-E5FC Hilfsroutine für Stringverkettung: String übertragen
- E5FD-E634 Entfernt Strings, die als „Zwischenwert“ bei einer Stringoperation entstanden sind (zum Beispiel im PRINT-Ausdruck)
- E635-E646 Korrektur der Deskriptorentabelle (als Folge der obengenannten Routine)
- E646-E659 CHR\$-Routine
- E65A-E685 LEFT\$-Routine
- E686-E690 RIGHT\$-Routine
- E691-E6B8 MID\$-Routine
- E6B9-E6D5 Holt Parameter für LEFT\$, RIGHT\$ und MID\$ vom Stack
- E6D6-E6E4 LEN-Routine
- E6E5-E6F4 ASC-Routine
- E6F5-E706 Holt 8-Bit-Zahl aus Basic-Text. Ist die Zahl größer als 255, dann „ILLEGAL QUANTITY ERROR“
- E707-E745 VAL-Routine
- E746-E751 Holt 16-Bit- und 8-Bit-Zahl aus Basic-Text. Nach Ausführung der Routine befindet sich der 16-bit-Wert in 50/51 und der 8-bit-Wert im Akkumulator
- E752-E763 Wandelt Inhalt von FAC 1 in Integer um. Ergebnis in 50/51
- E764-E77A PEEK-Routine
- E77B-E783 POKE-Routine
- E784-E79F WAIT-Routine
- E7A0-E7A6 Addiert 0.5 zu FAC 1.
- E7A7-E7B8 Subtrahiert Inhalt von FAC 1 von der „Variable“, deren Zeiger sich in A/Y befindet
- E7B9-E7C5 addiert Variable (Zeiger A/Y) zu FAC 1
- E7C6-E854 Addition FAC 1 + FAC 2. Ergebnis befindet sich wieder in FAC 1
- E855-E873 Addiert Mantissen von FAC 1 und FAC 2
- E874-E89D Normiert FAC 1: Mantisse wird so lange nach links verschoben, bis das höchstwertige Bit gesetzt ist. Das Exponentenbyte wird dabei entsprechend korrigiert
- E89E-E8D4 Mantisse von FAC 1 komplementieren (für Subtraktion und Addition zweier Zahlen mit ungleichem Vorzeichen)
- E8C2-E8D4 Erhöht FAC 1 um 1 in der letzten Bitstelle
- E8D5-E8D9 „OVERFLOW ERROR“ ausgeben
- E8DA-E912 Mantisse im FP-Zwischenspeicher (62 bis 65) nach rechts verschieben.
Einsprung E8F0: Verschiebt beliebiges Register (vier aufeinanderfolgende Bytes) um n Bits nach rechts. Adresse des höchstwertigsten Bytes minus 1 muß im X-Register stehen. Die Anzahl der Verschiebungen wird durch den Inhalt des Akkumulators bestimmt (256-n)
- E913-E917 FP-Konstante mit dem Wert 1
- E918-E92C Polynomkoeffizienten für LOG-Annäherung:
E918: 03 (Anzahl der Konstanten minus 1)
E919-E91D: 0,4342559419
E91E-E922: 0,5765845412
E923-E927: 0,9618007592
E928-E92C: 2,885390073
- E92D-E940 FP-Konstanten 0,5, 2, -0,5 und ln(2)
- E941-E97E LOG-Routine
- E97F-E9AF Multiplikation Variable (Zeiger A/Y) × FAC 1
- E987-E9AF Multiplikation FAC 1 × FAC 2. Ergebnis wiederum in FAC 1
- E9B0-E9E2 Multipliziert 8-Bit-Zahl im Akkumulator mit Argument von FAC 2. Aufsummierung der Teilprodukte im FP-Hilfsregister (62 und 65)
- E9E3-EA0D Überträgt Variable (Zeiger A/Y) nach FAC 2
- EA0E-EA2A Addiert Exponenten von FAC 1 und FAC 2 (für Multiplikation FAC 1 × FAC 2)
- EA2B-EA38 Überprüft Vorzeichen von FAC 1: Bei positivem Vorzeichen Fehlermeldung „OVERFLOW ERROR“, andernfalls wird FAC 1 auf Null gesetzt (Hilfsroutine für EXP-Berechnung)
- EA39-EA4F Multiplikation FAC 1 × 10
- EA50-EA54 FP-Konstante mit dem Wert 10
- EA55-EA65 Division FAC 1 / 10
- EA66-EACF Division Variable (Zeiger A/Y) durch FAC 1
- FA6B-EAE0 Division FAC 2 durch FAC 1
- EAE1-EAE5 „DIVISION BY ZERO ERROR“ ausgeben
- EAE6-EAF8 Überträgt Mantisse der FP-Hilfsvariable nach FAC 1
- EAF9-EB1D Überträgt Variable (Zeiger A/Y) nach FAC 1
- EB1E-EB52 Überträgt den gerundeten Wert von FAC 1 nach FAC 4. Löscht die Schutzstelle von FAC 1
- EB21-EB52 Überträgt FAC 1 nach FAC 3. Sonst wie oben
- EB27-EB52 Überträgt FAC 1 nach beliebige Stelle im Speicher. Die Zieladresse befindet sich dabei in 85/86
- EB53-EB62 Überträgt FAC 2 nach FAC 1 und löscht Schutzstelle von FAC 1
- EB63-EB71 Überträgt gerundeten Wert von FAC 1 nach FAC 2
- EB72-EB81 Rundet FAC 1
- EB82-EB8F Überprüft Vorzeichen von FAC 1. Bei negativem Vorzeichen enthält Akkumulator nach Ablauf der Routine den Wert 255, bei positivem Vorzeichen den Wert 1 und falls FAC 1 = 0 ist den Wert 0
- EB90-EBAE SGN-Routine
- EBAF-EBB1 ABS-Routine
- EBB2-EBF1 Vergleich zweier FP-Zahlen: vergleicht Variable (Zeiger A/Y) mit FAC 1
- EBF2-EC22 Umwandlung der FP-Zahl im FAC 1 in Integer
- EC23-EC49 INT-Routine
- EC4A-ED09 Umwandlung einer Zahl im ASCII-Format (String) in FP-Variable
- ED0A-ED18 Konstanten für Umwandlung FP nach String
ED0A-ED0E: 99 999 999,91
ED0F-ED13: 999 999 999,2
ED14-ED18: 1 000 000 000
- ED19-ED33 „LN“ mit Zeilennummer ausgeben (für Fehlermeldungen)

- ED24-ED33 16-Bit-Zahl in A/Y ausgeben
ED2E-ED33 FAC 1 ausgeben
ED34-EE63 Umwandlung der FP-Zahl im FAC 1 in ASCII-Format (String). Der neu entstandene String wird ab 100 abgelegt. Stringende ist durch ein Byte 00 gekennzeichnet. Nach Ablauf der Routine ist in A/Y die Anfangsadresse des Strings enthalten
- EE64-EE68 FP-Konstante mit dem Wert 0,5
EE69-EE8C Konstanten für Umwandlung FP nach String. Die Tabelle enthält den Wert der einzelnen Stellen des Dezimalsystems im Binärformat
- EE8D-EECF SQR-Routine (Berechnung über Potenzfunktion)
EE97-EECF Potenzfunktion FAC 2 \uparrow FAC 1
EED0-EEDA Vorzeichenwechsel
EEDB-EEDF FP-Konstante mit dem Wert 1,442695041 (1/ln2)
EEE0-EF08 Polynomkoeffizienten für EXP-Annäherung:
EEE0: 07 (Anzahl der Konstanten minus 1)
EEE1-EEE5: $2,150016068 \times 10^{-5}$
EEE6-EEEE: $1,435231404 \times 10^{-4}$
EEEE-EEEE: $1,342263482 \times 10^{-3}$
EEF0-EEF4: $9,614017013 \times 10^{-3}$
EEF5-EEF9: $5,550512686 \times 10^{-2}$
EEFA-EEFE: 0,2402263846
EEFF-EF03: 0,6931471862
EF04-EF08: 1
- EF09-EF5B EXP-Routine
EF5C-EF71 Berechnung eines Polynoms der Form
 $F(x) = A1 \times X + A3 \times X \uparrow 3 + A5 \times X \uparrow 5 + \dots$
A/Y dienen als Zeiger auf Tabelle mit den Polynomkoeffizienten. Das erste Byte der Tabelle muß die Anzahl der Glieder minus 1 beinhalten. Die Variable X muß in FAC 1 stehen. Die eigentliche Berechnung von F(x) erfolgt in der nachfolgenden Routine (Sprung nach E97F)
- EF72-EFA5 Berechnung eines Polynoms der Form
 $F(x) = A0 + A1 \times X + A2 \times X \uparrow 2 + A3 \times X \uparrow 3 + \dots$
sonst wie oben
- EFA6-EFAD Konstanten für RND-Funktion
EFAE-EFE9 RND-Routine
EFEA-FFF3 COS-Routine: Die Berechnung von COS-Werten wird auf die Berechnung von SIN-Werten zurückgeführt
- EFF4-F039 SIN-Routine
F03A-F065 TAN-Routine: Die Berechnung von TAN-Werten wird auf die Berechnung von SIN-Werten zurückgeführt
- F066-F06A FP-Konstante mit dem Wert 1,570796327 (Pi/2)
F06B-F06T FP-Konstante mit dem Wert 6,283185307 (Pi \times 2)
F070-F074 FP-Konstante mit dem Wert 0,25
F075-F093 Polynomkoeffizienten für SIN-Annäherung:
F075: 05 (Anzahl der Konstanten minus 1)
F076-F07A: -14,38139067
F07B-F07F: 42,00779712
F080-F084: -76,70417026
F085-F089: 81,60522369
F08A-F08E: -41,34170210
F08F-F093: 6,283185307 (2 \times Pi)
- F094-F098 Codes für „MICROSOFT!“
F099-F0CD ATN-Routine
F0CE-F105 Polynomkoeffizienten für ATN-Annäherung:
F0CE: 0B (Anzahl der Konstanten minus 1)
F0CF-F0D3: $-6,847939119 \times 10^{-4}$
F0D4-F0D8: $4,850942156 \times 10^{-3}$
F0D9-F0DD: $-1,611170184 \times 10^{-2}$
F0DE-F0E2: $3,420963805 \times 10^{-2}$
F0E3-F0E7: $-5,427913276 \times 10^{-2}$
F0E8-F0EB: $7,245719654 \times 10^{-2}$
F0EC-F0F1: $-8,980239538 \times 10^{-2}$
F0F2-F0F6: 0,1109324134
F0F7-F0FB: -0,1428398077
F0FC-F100: 0,1999991205
F101-F105: -0,3333333157
- F106-F10A FP-Konstante mit dem Wert 1
F10B-F122 Kopie der CHRGET-Routine (wird beim Kaltstart des Interpreters in die Zero-Page übertragen)
- F123-F127 FP-Konstante: Anfangswert der RND-Funktion (wird beim Kaltstart des Interpreters in RND-Register [C9 bis CD] übertragen)
- F128-F1D4 Basic-Kaltstart: Initialisierung des Interpreters. Feststellen der oberen Speichergrenze (HIMEM), Zeiger und Vektoren setzen
- F1D5-F1DD CALL-Routine
F1DE-F1E4 IN#-Routine: eigentliche Ausführung durch Monitor (Sprung nach FE8B)
F1E5-F1E9 PR#-Routine: eigentliche Ausführung durch Monitor (Sprung nach FE95)
- F1EC-F208 Holt PLOT-Parameter für LORES-Grafik. Ist eine der beiden Zahlen größer oder gleich 48, dann „ILLEGAL QUANTITY ERROR“. Nach Ablauf der Routine befindet sich die erste Zahl in F0 und die zweite in 2C und 2D
- F209-F224 Holt HLIN bzw. VLIN-Parameter. Die ersten beiden Zahlen werden wie oben in F0, 2C und 2D abgelegt. Ist die zweite Zahl größer als die erste, so werden sie vertauscht. Die dritte Zahl befindet sich nach Ablauf der Routine im X-Register
- F225-F231 PLOT-Routine: Ausführung durch Monitor (F800)
F232-F240 HLIN-Routine: Ausführung durch Monitor (F819)
F241-F24E VLIN-Routine: Ausführung durch Monitor (F828)
F24F-F255 COLOR = - Routine: Ausführung durch Monitor (F864)
F256-F261 VTAB-Routine: Ausführung durch Monitor (FB5B)
F262-F26C SPEED-Routine: Speichert Zweierkomplement zur nachfolgenden Zahl in F1 ab (für Verzögerungsschleife bei Ausgabe)
- F26D-F272 TRACE einschalten
F26F-F272 TRACE abschalten (NOTRACE)
F273-F27F NORMAL-Flags setzen
F277-F27F INVERSE-Flags setzen
F280-F285 FLASH-Flags setzen
F286-F2A5 HIMEM:-Routine: Beim Versuch den HIMEM-Wert unter die Endadresse der Feldvariablen zu setzen, erfolgt Sprung nach D410: „OUT OF MEMORY ERROR“
- F2A6-F2CA LOMEM:-Routine: ruft nach Ablauf D66C (CLR) auf
F2CB-F2E8 ONERR-Routine: Speichert CHRGET-Zeiger, der auf nachfolgenden GOTO-Befehl zeigt in F4/F5. Die aktuelle Zeilennummer wird nach F6/F7 übertragen, und der CHRGET-Zeiger wird auf die nächste Befehlszeile vorge-rückt
- F2E9-F317 Fehlerbehandlung bei aktivem ONERR GOTO: Der Error-Code wird in DE gespeichert und der Inhalt von 75/76 nach DA/DB übertragen (aktuelle Zeilennummer muß für ein RESUME abgespeichert werden). Der Inhalt von 79/7A wird nach DC/DD und der Stackpointer wird nach F8 gerettet. Danach Durchführung des ONERR-GOTO-Befehls
- F18-F32F RESUME-Routine
F33-F64 DEL-Routine
F365-F38F Löscht die Basic-Zeilen zwischen (9B/9C) und (60/61). Die hinter dem zu löschenden Bereich befindlichen Basic-Zeilen werden im Speicher nach unten verschoben
- F390-F396 GR-Routine: Ausführung durch Monitor (FB40)
F399-F39E TEXT-Routine: Ausführung durch Monitor (FB39)
F39F-F3BB STORE-Routine: speichert Länge des Feldes und Daten des Feldes nacheinander ab
- F3BC-F3D7 RECALL-Routine
F3D8-F410 HGR2-Routine: Setzt Flag für die zweite (HIRES-)Grafikseite. Löscht Bildschirm
- F3E2-F410 HGR-Routine: setzt Flag für die erste (HIRES-)Grafikseite und löscht Bildschirm
- F3F2-F410 Löscht aktuelle (zuletzt angesprochene) Grafikseite
F3F6-F410 Färbt die zuletzt angesprochene HGR-Seite mit der Farbe ein, deren Farbmaske (siehe F6F6-F6FD) sich in 1C befindet. Bei Aufruf dieser Routine muß Register E6 entweder 20 oder 40 beinhalten

- F411-F456 Berechnet die RAM-Adresse und Maske für das Zeichen eines Punktes in HGR aus X- und Y-Koordinate. Bei Aufruf muß die X-Koordinate in X/Y und die Y-Koordinate in A stehen. Nach Ablauf der Routine befindet sich die RAM-Adresse in 26/27 und die Maske in 30
- F457-F464 Zeichnet einen Punkt in HGR. RAM-Adresse in 26/27, Maske in 30
- F465-F49B Korrektur der RAM-Adresse und der Maske für Punkt links bzw. rechts neben altem Punkt. Für Bewegung nach rechts muß N-Flag gelöscht und für Bewegung nach links gesetzt sein
- F47C-F489 Falls Farbmaske eine alternierende Bitfolge besitzt, wird sie mit 7F Exklusiv-ODER-verknüpft. Dies ist notwendig, da die Anzahl der Punkte pro Byte, die auf dem Bildschirm dargestellt werden, ungerade ist
- F49C-F4B2 Unterprogramm für XDRAW-Routine: invertiert einen Punkt. RAM-Adresse in 26/27, Maske in 30
- F4B3-F4D4 Unterprogramm für DRAW-Routine: setzt einen Punkt. RAM-Adresse in 26/27, Maske in 30, Farbmaske in 1C
- F4D5-F503 Berechnet RAM-Adresse für Punkt über altem Punkt (Bewegung nach oben)
- F504-F52F Berechnet RAM-Adresse für Punkt unter altem Punkt (Bewegung nach unten)
- F530-F539 Löscht X- und Y-Koordinate in Register E0 bis E2. Diese Routine ermöglicht das Zeichnen beliebiger Linien in HGR mit relativer Adressierung. Sie geht in die folgende über
- F53A-F5B1 Zeichnen einer Linie in HGR. Alte X- und Y-Koordinate in E0/E1 bzw. E2, neue X-Koordinate in A/X und neue Y-Koordinate in Y. Nach Ablauf der Routine befinden sich neue Koordinaten in E0/E1 und E2
- F5B2-F5B8 Tabelle mit Masken für Plot-Routinen (siehe F411-F456)
- F5B9 Byte 1C für Vergleiche mit Hilfe des BIT-Befehls
- F5BA-F5CA Tabelle zur Bestimmung der Richtung eines Vektors bei verschiedenen ROT-Werten
- F5CB-F600 Berechnet X- und Y-Koordinate aus RAM-Adresse und Maske. Die Koordinaten werden in EO/E1 bzw. E2 abgelegt
- F601-F65C DRAW-Routine. Einsprung F605. Zeiger auf Definition der Shapes in 1A/1B
- F65D-F6B8 XDRAW-Routine. Einsprung: F661
- F6B9-F6E8 Holt Parameter für HPLOT, DRAW und XDRAW-Befehl aus Basic-Text. X-Koordinate nach Ablauf in X/Y, Y-Koordinate in A. Falls nötig, Ausgabe von „ILLEGAL QUANTITY ERROR“
- F6E9-F6F5 HCOLOR-Routine: Speichert Farbmaske in E4 ab
- F6F6-F6FD Tabelle mit Farbmasks für die Farben Schwarz (0), Grün (1), Violett (2), Weiß (3), Schwarz (4), Orange (5), Blau (6), Weiß (7)
- F6FE-F720 HPLOT-Routine
- F721-F726 ROT=- Routine
- F727-F72C SCALE=- Routine
- F72D-F768 DRAW-Parameter holen. Speichert Anfangsadresse des gewünschten Shapes in 1A/1B
- F769-F76E DRAW-Routine, Ausführung in F601-F65C
- F76F-F774 XDRAW-Routine, Ausführung in F65D-F6B8
- F775-F7BB SHLOAD-Routine
- F7BC-F7D8 Setzt Anfangs- und Endadresse der Daten eines Feldes für STORE und RECALL
- F7D9-F7E6 Sucht Feld für STORE und RECALL
- F7E7-F7FC HTAB-Routine
- F7FD-F7FF ???

Grafik mit dem MX-80

Woher nimmt man das achte Bit beim Apple?

Wenn Sie sich zu Ihrem Apple-II einen MX-80 gekauft und endlich alle Druckersteuerzeichen ausgeknobelt haben, kommt bald der Moment, an dem Sie Lust verspüren, die im Zeichenvorrat des MX-80 vorhandenen Grafiksymbole zu verwenden. Sie errechnen sich die nötigen CHR\$-Befehle und versuchen es. Aber Sie werden eine herbe Enttäuschung erleben.

Das für Grafiksymbole benötigte achte Bit ist auf der Interfaceplatine fest auf Masse gelegt, es ist also nicht ansteuerbar. Wenn Sie dann weitersuchen, werden Sie feststellen, daß die CHR\$-Befehle das achte Bit gar nicht ansteuern, obwohl CHR\$(128) bis CHR\$(255) durchaus erlaubt sind. Wie kann man dieses Bit trotzdem ansteuern?

Das Prinzip ist folgendes: Man verwendet einen TTL-Ausgang (AN2) des Game-Paddle-Anschlusses. Dazu ist lediglich eine kleine Hardwareänderung notwendig. Auf der Interfaceplatine befindet sich eine Drahtbrücke unterhalb des IC 3A (EPROM), bezeichnet mit P4. Sie

löten oder schneiden diese Drahtbrücke auf und verbinden das rechte Lötauge von M4 oder das linke Lötauge von P4 mit dem Game-Paddle-Anschluß AN2 (Pin 13). Der Anschluß auf der Interfaceplatine muß mit aller Sorgfalt hergestellt werden. Wenn Sie das falsche Lötauge erwischen, könnte im Computer das Ansteuer-IC des Game-Paddle-Anschlusses beschädigt werden. Prüfen Sie daher auf der Rückseite der Interfaceplatine, ob Sie wirklich ein Lötauge anschließen, das mit dem Stecker des Druckers verbunden ist. Die andern, „falschen“ Lötungen sind mit Masse bzw. IC 2A (74LS175) verbunden.

Wenn Sie diese Operation glücklich hinter sich gebracht haben, können Sie den Drucker mit POKE-16291,0 auf Grafiksymbolen und mit POKE-16292,0 auf alphanumerische Zeichen umschalten.


Hinweis: Normalerweise ist der TTL-Ausgang AN2 des Computers beim Einschalten auf Low (also Drucker im Normalbetrieb). Bei kurzzeitigem Ein- und Ausschalten des Computers kann er je-

doch auf High gehen, Sie müssen dann mit POKE-16292,0 den Drucker auf Normalbetrieb schalten.

Um das Blatt lückenlos mit Grafikzeichen bedrucken zu können, müssen Sie den Drucker auf den Zeilenabstand 7/72 Zoll umstellen (Befehl Esc 1 \triangleq CHR\$(27); CHR\$(49)). Die Grafikzeichen sind in einer 2x3-Matrix angeordnet. Sie können die Codes der Anleitung entnehmen. Es gibt aber auch einen einfacheren Weg, sie zu berechnen. Die Tabelle zeigt, wie man das macht.

Dieter Fischlin

Berechnung der MX-80-Grafikcodes

Position des Punktes	Wert
Oben links	1
Oben rechts	2
Mitte links	4
Mitte rechts	8
Unten links	16
Unten rechts	32
Grundwert	32
Beispiel:	 $1 + 8 + 16 + 32 = 57$ PRINT CHR\$(57)

Ernst Gülich

Shapemaker spart mühsame Kleinarbeit

Das folgende Programm vereinfacht die Erstellung von „Shapes“ beim Apple-II. Das sind frei definierbare Grafiksymbole, die man per Basic-Befehl dann in ihrer Lage und Größe ändern kann.

Der Apple II besitzt eine wunderschöne Eigenschaft: Im Grafikbetrieb lassen sich beliebige Symbole (sogenannte Shapes) definieren, die dann mit Basic-Befehlen vergrößert und gedreht werden können. Leider ist der Definitionsvorgang eine recht mühsame Angelegenheit. Das abgedruckte Programm „Shapemaker“ (Bild 1) befreit den Programmierer von der Kleinarbeit. Es fragt den Benutzer nach der Anzahl der Symbole und

der „Shape-Länge“. Die Symbole selbst werden durch Befehlskürzel (Tabelle) festgelegt. Sie werden ebenfalls der Reihe nach vom Programm angefordert. Bild 2 zeigt zwei Beispiele. In Zeile 1002 kann der Anfang der „Shapetable“ hinterlegt werden (abhängig von der Arbeitsspeichergöße). Das Programm setzt dann automatisch HIMEM und errechnet die Anzahl der Bytes, die auf Kassette oder Diskette abgespeichert

werden müssen. Wird die „Shapetable“-Startadresse geändert, sollte Zeile 1144 ebenfalls geändert werden: Die HEX-Anfangsadresse für das Abspeichern auf Kassette darf dann nicht mehr 4000 sein. Das Programm arbeitete im Dialog und läßt keine Falscheingaben zu. Vorsicht ist geboten beim Ändern des „Shapetable“-Anfangs, da das Programm die erste Grafikseite aufruft.

Befehle zum Erstellen eines Symbols (Shape)

Befehl	Wirkung
PL	Punkt setzen und nach links gehen
PR	Punkt setzen und nach rechts gehen
PU	Punkt setzen und nach oben gehen
PD	Punkt setzen und nach unten gehen
SL	Keinen Punkt setzen und nach links gehen
SR	Keinen Punkt setzen und nach rechts gehen
SU	Keinen Punkt setzen und nach oben gehen
SD	Keinen Punkt setzen und nach unten gehen

Wird dem Befehl die Zahl n nachgestellt, dann wird er n-mal ausgeführt.

```

1000 REM ***** SHAPEMAKER *****
1001 REM
1002 PP = 16384: REM # SHAPETABLEBEGIN #
1003 P1 = INT (PP / 256)
1004 P2 = INT ((PP / 256 - P1) * 256)
1005 HOME
1006 PRINT "SHAPE-MAKER WILL WORK FOR YOU"
1007 PRINT "*****"
1008 VTAB 5
1009 PRINT "NUMBER OF SHAPES MAX: 255"
1010 PRINT "LENGTH OF ONE SHAPE MAX: 255"
1011 PRINT "NORMAL"
1012 NORMAL : PRINT "SET POINT AND SHIFT DOWN=" : INVERSE : PRINT "PD"
1013 NORMAL : PRINT "SET POINT AND SHIFT LEFT=" : INVERSE : PRINT "PL"
1014 NORMAL : PRINT "SET POINT AND SHIFT RIGHT=" : INVERSE : PRINT "PR"
1015 NORMAL : PRINT "SET POINT AND SHIFT RIGHT=" : INVERSE : PRINT "PR"
1016 NORMAL : PRINT TAB(15);"SHIFT UP=" : INVERSE : PRINT "SU"
1017 NORMAL : PRINT TAB(15);"SHIFT DOWN=" : INVERSE : PRINT "SD"
1018 NORMAL : PRINT TAB(15);"SHIFT LEFT=" : INVERSE : PRINT "SL"
1019 NORMAL : PRINT TAB(15);"SHIFT RIGHT=" : INVERSE : PRINT "SR"
1020 NORMAL
1021 PRINT
1022 PRINT "IF A NUMBER(MAX.:99) IS FOLLOWING"
1023 PRINT "IT GIVES CODE*NUMBER"
1024 PRINT : PRINT
1025 INPUT TAB(4);"CONTINUE GIVE <GO> BREAK GIVE <END>"
1026 INPUT IN$: IF IN$ = "END" THEN HOME : PRINT "BYE": END
1027 IF IN$ < "GO" THEN 1000
1028 HOME : INPUT "GIVE NUMBER OF SHAPES :NR:"
1029 IF NR < 0 OR NR > 255 OR NR > INT (NR) THEN FLASH : GOTO 1000
1030 DIM N(NR)
1031 FOR I = 1 TO NR
1032 PRINT "GIVE LENGTH OF SHAPE :L:"
1033 IF L > 255 THEN L = 255: SJ = SJ + 1
1034 INPUT "SL(I) : "
1035 IF SL(I) < 0 OR SL(I) > 255 OR SL(I) > INT (SL(I)) THEN FLASH : CLEAR
1036 GOTO 1000
1037 NEXT I
1038 PRINT : INPUT "IT'S CORRECT <Y/N>":C$
1039 IF C$ = "N" THEN CLEAR : GOTO 1028
1040 IF C$ < "Y" THEN 1037
1041 POKE PP,NR: POKE PP+1,0
1042 POKE PP+2,NR * 2 + 2: POKE PP+3,0
1043 SJ = 0
1044 IF NR = 1 THEN 1050
1045 FOR I = 2 TO NR
1046 YV = INT ((SL(I) + SJ) / 256)
1047 XA = INT ((SL(I) + SJ) / 256 - YV) * 256
1048 POKE PP+2 * I+XX: POKE PP+2 * I+1,YV
1049 SJ = SJ + SL(I)
1050 NEXT I
1051 POKE 115,P2: POKE 116,P1
1052 POKE 232,P2: POKE 233,P1
1053 NAME IN$(255)
1054 PRINT "CORRECTION GIVE <C>"
1055 PRINT : INPUT "END OF SHAPE GIVE <E>"
1056 PRINT : INPUT "SHAPE NR.:JJ + 1"
1057 PRINT
1058 FOR I = 0 TO SL(JJ+1)
1059 PRINT "LINE=" : TAB(10); INPUT "":IN$(I)
1060 I = 0
1061 IF IN$(I) = "C" THEN GOSUB 1156
1062 IF I = 1 THEN 1060
1063 IF IN$(I) = "E" THEN 1078
1064 I$ = IN$(I)
1065 IN$(I) = LEFT$(IN$(I),2)
1066 IF I$ < "SL" AND (I$ < "SR") AND (I$ < "SU") AND (I$ < "SD")
1067 AND (I$ < "PL") AND (I$ < "PR") AND (I$ < "PU") AND (I$ < "PD")
1068 THEN 1060
1069 IF I = 1 THEN 1060
1070 ZA = VAL (MID$(I$,3,3))
1071 IF ZA = 0 OR ZA = 1 THEN 1075
1072 FOR ZA = ZA TO 2 STEP -1
1073 I = I + 1: IN$(I) = I$

```



```

1074 NEXT ZA
1075 NEXT I
1076 PRINT "END OF SHAPELENGTH"
1077 IN$(SL(JJ + 1) + 1) = "E"
1078 PRINT : INVERSE : PRINT " PLEASE WAIT " : NORMAL
1079 JJ = JJ + 1
1080 DY = 0
1081 ZZ = 0
1082 J1 = 0
1083 J1 = -1
1084 FOR J = 0 TO 1
1085 J1 = J1 + 1
1086 IF IN$(J) = "E" THEN DX = 0: Q = 1
1087 IF IN$(J) = "SL" THEN DX = 3
1088 IF IN$(J) = "SR" THEN DX = 1
1089 IF IN$(J) = "SD" THEN DX = 2
1090 IF IN$(J) = "SU" THEN DX = 0
1091 IF IN$(J) = "PL" THEN DX = 7
1092 IF IN$(J) = "PD" THEN DX = 6
1093 IF IN$(J) = "PR" THEN DX = 5
1094 IF IN$(J) = "PU" THEN DX = 4
1095 IF (DX = 0 AND J1 = 2) OR (DX = 6 AND J1 = 2) OR (DX = 7 AND J1 = 2) THEN DX = 0: J = J - 1
1096 OR (DX = DY + DX * 8 ^ J1
1097 IF J1 = 2 THEN J1 = -1: GOTO 1099
1098 NEXT J
1099 IF DY = 0 THEN GOSUB 1167
1100 POKE PP + PEEK (PP + 2 * JJ) + PEEK (PP + 2 * JJ + 1) * 256 + ZZ +
JJ, DY
1101 DY = 0
1102 ZZ = ZZ + 1
1103 IF Q = 1 THEN 1105
1104 NEXT J
1105 POKE PP + PEEK (PP + 2 * JJ) + PEEK (PP + 2 * JJ + 1) * 256 + ZZ +
JJ, DY
1106 GOSUB 1172
1107 IF JJ < NR THEN 1053
1108 HOME
1109 PRINT "DO YOU LIKE SAVE AT"
1110 PRINT " CASSETTE GIVE <C>"
1111 PRINT " DISK "
1112 PRINT " NOSAVE GIVE <N>"
1113 Q = 0
1114 PRINT : INPUT " :LI$
1115 IF LI$ = "C" THEN GOSUB 1123
1116 IF LI$ = "D" THEN GOSUB 1147
1117 IF LI$ = "N" THEN 1120
1118 IF Q = 1 THEN 1120
1119 HOME : FLASH : GOTO 1109
1120 NORMAL : PRINT
1121 PRINT " IT'S DONE"
1122 END
1123 M = PEEK (PP)
1124 N = PEEK (PP + M * 2) + PEEK (PP + M * 2 + 1) * 256
1125 LE = N + SL(M)
1126 START = PP
1127 BREAK = START + LE
1128 POKE 0, (LE / 256 - INT (LE / 256)) * 256: POKE 1, INT (LE / 256)
1129 HOME : NORMAL
1130 PRINT "CALL MONITOR (CALL -151) AND GIVE:"
1131 FOR I = 0 TO 3
1132 ZX(I) = (BREAK / 16 - INT (BREAK / 16)) * 16
1133 BREAK = INT (BREAK / 16)
1134 NEXT I
1135 DATA "A", "B", "C", "D", "E", "F"
1136 FOR I = 0 TO 3: ZX(I)
1137 IF ZX(I) < 10 THEN X$ = STR$(ZX(I)): GOTO 1142
1138 READ X$
1139 NEXT I
1140 NEXTORE
1141 ZX$(1) = X$
1142 PRINT "
1143 PRINT " 0.1W 4000. :ZX$(3):ZX$(2):ZX$(1):ZX$(0):" :W"
1144 RETURN
1145 HOME
1146 INPUT "GIVE NAME FOR SHAPETABLE " : NA$
1147 M = PEEK (PP)
1148 N = PEEK (PP + M * 2) + PEEK (PP + M * 2 + 1) * 256

```

Bild 1. Listing des Programms „Shapemaker“

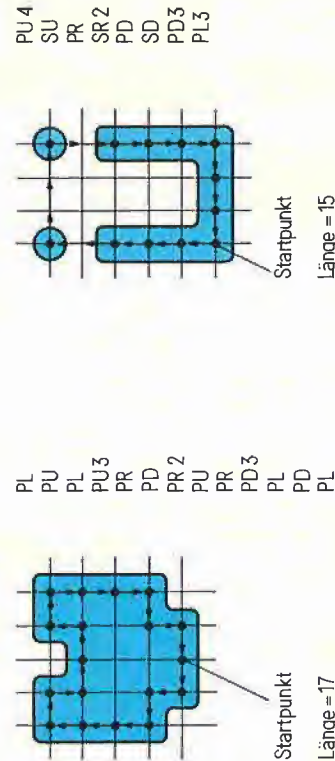


Bild 2. Zwei Symbole mit den zugehörigen Befehlssequenzen: Der Startpunkt ist wichtig, wenn die Figur gedreht werden soll. Die Länge, die das Programm erfragt, ergibt sich aus der Anzahl der „Zeichenschritte“ minus eins

Rudolf Hofer

Mit MX-82 und Apple:

Strichcode drucken und lesen

Der mc-Strichcode eignet sich nicht nur zur Übertragung von Programmen. Zahlreiche Leser verwenden ihn – ähnlich wie in Supermärkten – zur Kennzeichnung von Artikeln. Auch zur Auswertung von Schülerarbeiten bietet er sich an. Für derartige Anwendungen sind die nachfolgend vorgestellten Programme gedacht. Das erste, ein Basic-Programm, druckt numerische oder alphanumerische Daten im Strichcode. Beim zweiten handelt es sich um ein Maschinenprogramm, das diese Daten liest, auf Fehler prüft und einer Variablen zuweist.

Die Zahl der Zeichen ist einstellbar

Das im Bild 1 dargestellte Programm druckt Etiketten, die im mc-Strichcode dargestellt sind. Im Unterschied zu Programmausdrucken [2] wird jedoch keine Länge angegeben, und die Prüfsumme besteht nur aus einem Zeichen (Bild 2). Das Leseprogramm muß deshalb darauf eingestellt werden. Die Anordnung der Strichcodewörter auf dem Papier ist in weiten Grenzen wählbar. So können mehrere Etiketten in einer Zeile gedruckt werden, der Abstand (in Punkten) ist in Zeile 8002 mit der Variablen ZW einzustellen, und der vertikale Abstand kann vergrößert werden, indem man in Zeile 8056 mehrere Male „Line Feed“ (CHR\$10) ausgibt.

ZW ist mit Null vorbesetzt. Das bedeutet in der Praxis, daß bei maximaler Etikettenlänge (lauter Einsen) das nächste Etikett ohne Abstand folgt. Das dieser Fall aber höchstens theoretisch auftritt, gibt es keine Probleme. Soll auf Etikettenträger gedruckt werden, dann ermittelt man den richtigen Wert von ZW (damit an

```

8000 REM *****HAUPTPROGRAMM*****
8002 ZW = 0: REM ZWISCHENRAUM
8006 INPUT "ALPHA/NUMERISCH (1/0): ";ALZ
8008 INPUT "TEXT: ";A$:N = LEN (A$)
8010 NP = 80 * (N + 1) + 20 + ZW:AZ(N) = 0
8012 IF ALZ = 0 THEN NP = 40 * (N + 1) + 20 + ZW
8014 INPUT "WIE OFT DRUCKEN: ";DW
8016 IF DW * NP > 960 THEN 8014
8018 IF ALZ = 0 THEN 8024
8020 FOR I = 1 TO N:AZ(I - 1) = ASC ( MID$ (A$,I,1)): NEXT I
8022 GOTO 8026
8024 FOR I = 1 TO N:AZ(I - 1) = VAL ( MID$ (A$,I,1)): NEXT I
8026 D$ = CHR$ (4): POKE 54,0: POKE 55,3: CALL 1002
8028 DIM AB$(NP / 2): REM *****IM HAUPTPROGRAMM DIM.***
8030 GOSUB 8082: REM ZEILE BERECHNEN
8032 N2 = INT (NP / 256):N1 = NP - 256 * N2
8034 FOR LK = 1 TO 3: FOR LL = 1 TO 2: FOR LW = 1 TO DW
8036 PRINT CHR$ (27);"L";
8038 IF N1 > 127 THEN POKE 769,255
8040 PRINT CHR$ (N1);: POKE 769,127
8044 PRINT CHR$ (N2);
8046 GOSUB 8106: NEXT LW: REM EINE ZEILE
8048 PRINT CHR$ (27)"A" CHR$ (7): PRINT : NEXT LL
8049 PRINT CHR$ (10)
8050 PRINT CHR$ (27)"2": NEXT LK
8051 PRINT CHR$ (10)
8052 PRINT : FOR LW = 1 TO DW
8054 PRINT A$: SPC( NP / 12 - N + 1): NEXT LW
8056 PRINT CHR$ (10): REM ZEILENABSTAND
8058 PRINT : PRINT D$"PR#0"
8060 END
8062 REM *****EINE ZEILE*****
8064 MZ = 1: REM MODULZAHLER
8066 FOR I = 0 TO NP / 2:AB$(I) = 0: NEXT I
8068 GOSUB 8076: REM SYNCH-BALKEN
8070 FOR L = 0 TO N - 1: GOSUB 8088:AZ(N) = AZ(N) + AZ(L): NEXT L
8072 L = N: GOSUB 8088: REM PRÜFSUMME
8074 RETURN
8076 REM *****SYNCH*****

```

```

8078 GOSUB 8082: REM ZWEIMAL BREIT
8080 REM -----BREIT-----
8082 AB$(MZ) = 255:AB$(MZ + 1) = 255:MZ = MZ + 2
8084 REM -----SCHMAL-----
8086 AB$(MZ) = 255:MZ = MZ + 3: RETURN
8088 REM *****EIN ZEICHEN*****
8090 CHZ = AZ(L)
8092 FOR I = 1 TO 8
8094 IF 2 * INT (CHZ / 2) < > CHZ THEN GOSUB 8082: GOTO 8098
8096 GOSUB 8086
8098 CHZ = CHZ / 2
8100 IF ALZ = 0 AND I = 4 THEN 8104
8102 NEXT I
8104 RETURN
8106 REM *****DRUCKEN*****
8108 FOR I = 1 TO NP / 2: PRINT CHR$ (AB$(I));: PRINT CHR$ (AB$(I));: NEXT I
8110 RETURN

0800      1 ; ZEICHEN AUSGEBEN
0800      2 ; AUF DRUCKER
0800      3 ; STATT MIT PR#1
0800      4 ; WIRD DER DRUCKER
0800      5 ; MIT POKE 54,0:POKE55,3
0800      6 ; (UND DANACH CALL1002,
0800      7 ; FALLS DOS VORHANDEN)
0800      8 ; INITIALISIERT.
0800      9 ; IN ADR.305 STEHT C2,
0800     10 ; UND IN 30A STEHT A0,
0800     11 ; FALLS INTERFACE IN
0800     12 ; SLOT 2 STECKT USW.
0800     13 ; -----
0300     14      ORG $300
0300     15      OBJ $800
0300    297F     16      START AND #$7F
0302     48      17      PHA
0303    2CC1C1    18      LOOP BIT $C1C1
0306    30FB     19      BMI LOOP
0308     68      20      PLA
0309    8D90C0    21      STA $C090
030C     60      22      RTS
030D           23      PAU

```

Bild 1. Das Druckprogramm: Bevor es gestartet wird, muß das kleine Maschinenprogramm geladen werden. Es gibt ein Zeichen über das Epson-Interface (Slot 1) an den Drucker aus. Falls ein Eigenbau-Interface verwendet wird, steht ab \$300 die Routine zur Ausgabe eines Zeichens. Wichtig dabei: Der erste Befehl muß AND#\$7F sein, da das Basic-Programm das zweite Byte als Maske verwendet

der richtigen Stelle neu begonnen wird) durch Probieren.
Falls man mehrere Seiten an Aufklebern drucken will, ist das Programm als Subroutine auszulegen und von einem Hauptprogramm her entsprechend oft aufzurufen. Dabei empfiehlt es sich, die Parameter DW (Etiketten pro Zeile) und AL% (alphanumerisch/numerisch = 1/0) vorher zu definieren und die Zeilen 8006 sowie 8014 wegzulassen. Der DIM-Befehl in Zeile 8028 muß dann ebenfalls im Hauptprogramm (vor der Ausgabeschleife) stehen. END in Zeile 8060 ist durch RETURN zu ersetzen. Als Parameter wird dem Unterprogramm der String A\$ übergeben, der die auszugebende Zeichenfolge enthält. Führende Nullen sind mit anzugeben, falls man mit fester Zeichenzahl arbeitet.

Die Länge der Balken wird von der Laufvariablen LK in Zeile 8034 vorgegeben. Sie ist auf drei Durchläufe eingestellt. Jeder weitere Durchlauf verlängert die Balken um knappe drei Millimeter. Beim Drucker darf die Auto-Feed-Funktion (aut. LF nach CR) nicht aktiviert sein. Andernfalls müssen die Print-Befehle in den Zeilen 8048...8051 angepaßt werden.

Damit ein einwandfreier Lesebetrieb gewährleistet ist, muß das Farbband noch genügend Kontrast liefern.

Das Leseprogramm: mit wenig Aufwand anzupassen

Das Maschinenprogramm in Bild 3 liest eine einstellbare Anzahl von Zeichen und weist sie beim folgenden Input-Befehl einer Variablen zu. Es besteht aus einem Initialisierungsteil und dem eigentlichen Leseprogramm. Im Initialisierungsteil wird der Wert von HIMEM auf \$9300 eingestellt und die Eingabe von der Tastatur auf den Strichcodeleser umgelenkt. Außerdem modifiziert sich das Programm selbst. Und zwar so, daß abhängig von der Einsprungsadresse entweder 8 Bit oder 4 Bit pro Zeichen gelesen werden. Aus diesem Grund kann es nur im RAM ablaufen.

Folgende Adressen sind für den Anwender von Bedeutung (in Klammern jeweils die dezimalen Werte):

- 9300 (37632): Einsprungsadresse für numerische Daten
- 930E (37646): Einsprungsadresse für ASCII-Zeichen

- 932B (37675): HIMEM-Page; für HIMEM=\$9000 würde hier der Wert 90 stehen
- 933A...C: =EA EA EA, falls kein Diskettenlaufwerk angeschlossen ist
- 936C (37740): Zahl der einzulesenden Zeichen (ohne Prüfsumme)
- 93D4 (37844): Lesegeschwindigkeit (Wert auf „flotten“ Betrieb eingestellt); bei einem größeren Wert muß langsamer gelesen werden
- 940B...: Puffer für eingelesene Zeichen + CR

Das Programm läuft auf einem 48-KByte-Apple. Bild 4 zeigt, wie es von Basic aus angesprochen wird. In Zeile 5 wählt man die Anzahl der einzulesenden Zeichen (ohne Prüfzeichen). Zeile 10 ruft den Initialisierungsteil für alphanumerische (ASCII) Zeichen auf. Wenn man numerische Daten lesen will, muß an dieser Stelle CALL 37632 stehen. Sollen immer Daten mit gleicher Stellenzahl gelesen werden, speichert man das Programm ab, nachdem es einmal abgelaufen ist. Zeile 5 ist dann nicht mehr erforderlich.

DRUCKER-SLOT: 2
ALPHA/NUMERISCH (1/0): 0
TEXT: 1234
WIE OFT DRUCKEN: 2



DRUCKER-SLOT: 2
ALPHA/NUMERISCH (1/0): 1
TEXT: ABCDE
WIE OFT DRUCKEN: 1



Bild 2. Beispielausdrucke: Im ersten Fall wurde die ASCII-Darstellung gewählt, im zweiten die numerische, die wesentlich weniger Platz in Anspruch nimmt. Die Prüfsumme wird jeweils durch Aufsummieren der Datenbytes bzw. Halbbytes ohne Übertrag gewonnen. Vorangestellt sind zwei breite Balken zur Synchronisation. Das niederwertige Bit steht jeweils am Anfang

Die im Strichcode verschlüsselten Daten werden in Zeile 20 direkt der Variablen A\$ zugewiesen, als wären sie von der Tastatur aus eingegeben worden. Bei numerischen Daten kann auch eine Integer- oder Gleitkomma-Variable verwendet werden. Übrigens passiert bei einem Lesefehler nichts. Erst wenn die Prüfsumme stimmt, erscheinen die Daten auf dem Bildschirm. Zur Bestätigung piepst dann der Apple-Lautsprecher. Zeile 25 sorgt dafür, daß die nächste Eingabe wieder von der Tastatur entgegengenommen wird.

Literatur

- [1] Hofer, Rudolf: Apple-II liest Strichcode. mc 1981, Heft 1, S. 42.
- [2] Wie ist der mc-Strichcode aufgebaut? mc 1981, Heft 1, S. 44.
- [3] Lesestift für mc-Programme. mc 1981, Heft 1, S. 45.

```

9300-- A9 04 8D E9 93 A9 80 8D
930B-- 7E 93 A9 4A D0 0C A9 0B
9310-- 8D E9 93 A9 80 8D 7E 93
931B-- A9 EA 8D F7 93 8D F8 93
9320-- 8D F9 93 8D FA 93 A9 00
932B-- 85 73 A9 93 85 74 A9 FF
9330-- 85 FE A9 3E 85 38 A9 93
933B-- 85 39 20 EA 03 60 86 F6
9340-- 24 FE 30 0E A6 FF E4 FA
934B-- F0 0B 8D 0B 94 E6 FF A6
9350-- F6 60 20 4A FF 2C 61 C0
935B-- 30 F8 20 FC 93 86 F8 86
9360-- F7 46 F7 20 FC 93 20 AD
936B-- 93 90 E7 A9 04 85 FA A2
9370-- 00 A9 00 85 FB 86 FD 20
937B-- E7 93 A6 FD 4B 09 80 9D
9380-- 0B 94 68 1B 65 FB 85 FB
938B-- E8 E4 FA D0 E8 20 E7 93
9390-- C5 FB D0 50 A9 8D A6 FD
939B-- E8 9D 0B 94 E6 FA 20 DD
93A0-- FB 20 3F FF A2 00 86 FF
93AB-- 86 FE 4C 46 93 A5 F7 4A
93B0-- 65 F7 85 F9 E4 F9 10 09
93BB-- 66 F7 A5 F7 0A 85 FB 1B
93C0-- 60 86 F8 A5 F8 4A 85 F7
93CB-- 3B 60 4B A2 00 2C 61 C0
93D0-- 10 FB E8 A9 1B E9 01 D0
93DB-- FC E0 FF F0 25 2C 61 C0
93E0-- 30 F0 68 60 4C 52 93 4B
93EB-- A0 0B 20 CA 93 20 AD 93
93F0-- 68 6A 4B 8B D0 F4 68 EA
93FB-- EA EA EA 60 4B 20 CA 93
9400-- 68 60 68 68 68 68 68
940B-- 4C 52 93
    
```

Bild 3. Leseprogramm: Das Assemblerlisting eines annähernd identischen Programms findet sich in [1]

```

5 POKE 37740,4: REM VIER STELLEN
15 CALL 37646: REM ALPHANUM.
20 INPUT "STRICHCODE LESEN: ";A$
25 PRINT CHR$(4)"IN#0"
    
```

Bild 4. So wird das Leseprogramm von Basic aus aufgerufen

Rudolf Hofer

Apple-II liest Strichcode

Um Programme (z. B. in Basic), die im Strichcode abgedruckt sind, in den Computer zu bringen, sind zwei Dinge nötig: ein Strichcode-Leser und ein kleines Hilfsprogramm.

Wer sich nicht dafür interessiert, wie das Programm in *Bild 1* im einzelnen funktioniert, kann es einfach anhand folgender Anleitung benutzen:

- ☐ Der Leser wird nach *Bild 2* an den sogenannten „Game I/O Connector“ des Apple angeschlossen.
- ☐ Die Daten aus *Bild 1* (A9 FF 85...) werden zeilenweise mit Hilfe des Apple-Monitors ab Adresse 3E00 eingetippt und auf Band bzw. Diskette gespeichert. Falls ohne Diskettenbetriebssystem gearbeitet wird, ist das Byte an der Stelle 3E0C (20) durch 60 zu ersetzen.
- ☐ Sollen Daten eingelesen werden, wird folgendes Kurzprogramm gestartet (nicht im Direktmodus eingeben):
10000 HOME
10001 HIMEM: 15871
10002 CALL 15872
Ab jetzt ist keine Eingabe von der Tastatur aus mehr möglich. An ihre Stelle tritt der Strichcodeleser.
- ☐ Die Strichcodezeilen werden der Reihe nach mit dem Leser abgefahren. Achtung: Etwas vor dem ersten Strich beginnen und nicht vor dem letzten Strich abstoppen! Leser u. U. an Lineal anlegen! Mit etwas Übung geht es aber frei Hand wesentlich flotter. Damit man jedoch mit geringer Geschwindigkeit noch lesen kann, muß die Zeitkonstante in Speicherzelle 3EA7 etwa auf den Wert 19 (hex) geändert werden. Ist eine Zeile richtig gelesen, erscheint der entsprechende Text auf dem Bildschirm, und man

kann die nächste Zeile einlesen. Liegt ein Lesefehler vor, dann gibt der Lautsprecher des Apple einen kurzen Piepton ab. In diesem Fall muß der Lesevorgang wiederholt werden. Sind alle Zeichen gelesen, drückt man die Reset-Taste, und die Tastatur ist wieder in Betrieb. Jetzt kann das Programm auf Kassette oder Diskette abgespeichert werden.

Funktion des Einleseprogramms

Das Programm beginnt bei Adresse 3E00 (hex), also am Ende des 16-K-Bereichs. Der vorher erwähnte Befehl CALL 15872 entspricht dem Start an der Stelle INIT. Dieser Programmteil ändert den Inhalt der Speicherzellen 38/39 (hex.), in denen normalerweise die Anfangsadresse der Tastaturabfrageroutine steht. Der neue Wert ist die Anfangsadresse des eigentlichen Einleseprogramms *READER*, dessen Ablauf in *Bild 3* dargestellt ist. Der Befehl JSR \$3EA ist nur mit Diskettenbetriebssystem erforderlich und muß ansonsten entfallen.

0800	1	; CODEREADER	
0800	2	; *****	
0800	3	XTEMP1 EPZ \$F6	
0800	4	REF0 EPZ \$F7	
0800	5	REF1 EPZ \$F8	
0800	6	REF EPZ \$F9	
0800	7	BUF EPZ \$FA	
0800	8	CHCKL EPZ \$FB	
0800	9	CHCKH EPZ \$FC	
0800	10	XTEMP EPZ \$FD	
0800	11	PFFLAG EPZ \$FE	
0800	12	PNT EPZ \$FF	
0800	13	KSWL EPZ \$38	
0800	14	KSWH EPZ \$39	
0800	15	PA EQU \$C061	; PORT
0800	16	ZEIT EQU 8	
0800	17	BELL1 EQU \$FBDD	
0800	18	IOSAVE EQU \$FF4A	
0800	19	IOREST EQU \$FF3F	
0800	20	; *****	
3E00	21	ORG \$3E00	
3E00 A9FF	22	INIT LDA ##FF	; FLAG PUFFER LEER
3E02 85FE	23	STA PFFLAG	
3E04 A910	24	LDA #READER	
3E06 8538	25	STA KSWL	
3E08 A93E	26	LDA /READER	
3E0A 8539	27	STA KSWH	
3E0C 20EA03	28	JSR \$3EA	; NUR MIT DOS
3E0F 60	29	RTS	

Bild 1. Listing des Strichcode-Einleseprogrammes für den Apple II. Es ist weitgehend modular aufgebaut. Das Programm nutzt die Tastaturabfrage-Routine des Apple II aus

Anstatt von der Tastatur bekommt der Computer also jetzt von diesem Programm Zeichen für Zeichen geliefert. Es muß lediglich dafür gesorgt werden, daß der Akku das gewünschte Zeichen enthält, daß am Ende der Befehl RTS (Return from Subroutine) steht und daß keine anderen Registerinhalte zerstört werden.

Da man die Daten vom Strichcodeleser erst auf ihre Richtigkeit prüfen muß, bevor man sie auf die beschriebene Weise an die „Zeichenschnittstelle“ des Computers übergibt, muß man sie erst in einem Puffer ablegen. Dieser Puffer wird so lange neu gefüllt, bis die Prüfsumme stimmt. Erst dann liegen lauter richtige Daten vor. Jetzt kann Zeichen für Zeichen übergeben werden.

Leider ist nicht bei jedem Computer eine solche Zeichenschnittstelle so problemlos zugänglich. Deshalb ist das Rahmenprogramm READER nicht universell verwendbar. Prinzipiell für jede 6502-Maschine ist aber der Teil geeignet, der im Flußdiagramm mit „Zeichen in Puffer einlesen bis o. k.“ bezeichnet ist. Das Programmsegment ab START liest eine Strichcodezeile und legt die Daten in einem Puffer ab. Es ruft verschiedene Unterprogramme auf und berechnet die Prüfsumme.

Aus dem Apple-Monitor werden BELL1 (Piepton erzeugen) IOSAVE (CPU-Register retten) und IOREST (CPU-Register wieder laden) verwendet. Das Unterprogramm BYTE liest ein Byte ein. Es ruft seinerseits BITTST und AUSW auf. Die letzten beiden Routinen sind der Kern des gesamten Einleseprogramms.

Die wichtigsten Unterprogramme

Die Routine BITTST ist leicht erklärt. Sie zählt im Indexregister X die Dauer eines Schwarzpegels – im allgemeinen die Strichstärke. Liegt der Leser nicht auf weißem Papier, liefert er ebenfalls „Schwarzpegel“. In diesem Fall fällt der Vergleich des Indexregisterinhaltes mit 255 irgendwann positiv aus, und der Prozessor springt nach UEBRL und beginnt wieder bei START. Die sechs PLA-Befehle korrigieren den Stackpointer, da in eine andere Programmebene gesprungen wird.

Die Routine AUSW, die immer nach BITTST aufgerufen wird, stellt fest, ob ein gelesener Strich eine „0“ oder

3E10	30	*****
3E10 86F6	31	READER STX XTEMP1
3E12 24FE	32	BIT PFLAG ; PUFFER LEER?
3E14 300E	33	BMI START ; PUFFER FUELLEN
3E16 A6FF	34	LDX PNT
3E18 E4FA	35	EING CPX BUF
3E1A F008	36	BEQ START
3E1C BDDD3E	37	LDA BUFFER, X
3E1F E6FF	38	INC PNT
3E21 A6F6	39	LDX XTEMP1
3E23 60	40	RTS
3E24	41	*****
3E24 204AFF	42	START JSR IOSAVE
3E27 2C61C0	43	BIT PA
3E2A 30F8	44	BMI START ; WARTEN AUF WEISS
3E2C 20CE3E	45	JSR SYNCH ; SYNCH ZAEHLEN
3E2F 86F8	46	STX REF1
3E31 86F7	47	STX REFO
3E33 46F7	48	LSR REFO
3E35 20CE3E	49	JSR SYNCH ; 2. SYNCH
3E38 20B03E	50	JSR AUSW
3E3B 907A	51	BCC FEHLER
3E3D 20BD3E	52	JSR BYTE
3E40 85FA	53	STA BUF ; PUFFERENDE
3E42 A200	54	LDX #0
3E44 A900	55	LDA #0
3E46 85FB	56	STA CHCKL
3E48 85FC	57	STA CHCKH
3E4A 86FD	58	VOR STX XTEMP
3E4C 20BD3E	59	JSR BYTE
3E4F A6FD	60	LDX XTEMP
3E51 48	61	PHA
3E52 0980	62	ORA ##80
3E54 9DDD3E	63	STA BUFFER, X
3E57 68	64	PLA
3E58 18	65	CLC
3E59 65FB	66	ADC CHCKL
3E5B 85FB	67	STA CHCKL
3E5D 9002	68	BCC NULL
3E5F E6FC	69	INC CHCKH
3E61 E8	70	NULL INX
3E62 E4FA	71	CPX BUF
3E64 D0E4	72	BNE VOR
3E66 20BD3E	73	JSR BYTE
3E69 C5FB	74	CMP CHCKL
3E6B D04A	75	BNE FEHLER
3E6D 20BD3E	76	JSR BYTE
3E70 C5FC	77	CMP CHCKH
3E72 D043	78	BNE FEHLER
3E74 203FFF	79	JSR IOREST
3E77 A200	80	LDX #0
3E79 86FF	81	STX PNT ; POINTER=0
3E7B 86FE	82	STX PFLAG ; PUFFER VOLL
3E7D 4C183E	83	JMP EING
3E80	84	*****
3E80	85	BIT AUSWERTEN
3E80 A5F7	86	AUSW LDA REFO
3E82 4A	87	LSR
3E83 65F7	88	ADC REFO
3E85 85F9	89	STA REF ; REFO+1/2
3E87 E4F9	90	CPX REF
3E89 1009	91	BPL EINS
3E8B 86F7	92	STX REFO ; BIT IST NULL
3E8D A5F7	93	LDA REFO
3E8F 0A	94	ASL
3E90 85F8	95	STA REF1
3E92 18	96	CLC ; CARRY IST NULL
3E93 60	97	RTS
3E94 86FB	98	EINS STX REF1
3E96 A5FB	99	LDA REF1
3E98 4A	100	LSR
3E99 85F7	101	STA REFO
3E9B 38	102	SEC ; CARRY IST EINS
3E9C 60	103	RTS ; BIT STEHT IN CARRY
3E9D	104	*****
3E9D 48	105	BITTST PHA
3E9E A200	106	LDX #0
3EA0 2C61C0	107	BIT BIT PA
3EA3 10FB	108	BPL BIT
3EA5 E8	109	ZAEHL INX

```

3EA6 A908 110 LDA #ZEIT
3EA8 E901 111 VERZ SBC #1
3EAA D0FC 112 BNE VERZ
3EAC E0FF 113 CPX #255
3EAE F024 114 BEQ UEBRL
3EB0 2C61C0 115 BIT PA
3EB3 30F0 116 BMI ZAEHL
3EB5 68 117 PLA
3EB6 60 118 RTS
3EB7 119 ; *****
3EB7 20DDFB 120 FEHLER JSR BELL1
3EBA 4C243E 121 JMP START
3EBD 122 ; *****
3EBD 123 ; BYTE EINLESEN
3EBD 48 124 BYTE PHA
3EBE A008 125 LDY #8
3EC0 209D3E 126 NBIT JSR BITTST
3EC3 20803E 127 JSR AUSW
3EC6 68 128 PLA
3EC7 6A 129 ROR
3EC8 48 130 PHA
3EC9 88 131 DEY
3ECA D0F4 132 BNE NBIT
3ECC 68 133 PLA
3ECD 60 134 RTS ; BYTE STEHT IN AKKU
3ECE 135 ; *****
3ECE 48 136 SYNCH PHA
3ECF 209D3E 137 JSR BITTST
3ED2 68 138 PLA
3ED3 60 139 RTS
3ED4 140 ; *****
3ED4 68 141 UEBRL PLA
3ED5 68 142 PLA
3ED6 68 143 PLA
3ED7 68 144 PLA
3ED8 68 145 PLA
3ED9 68 146 PLA
3EDA 4C243E 147 JMP START
3EDD 00 148 BUFFER BRK ; PUFFERANFANG
149 END

```

eine „1“ repräsentiert. Ein Maß für die Strichstärke ist der Inhalt des Indexregisters X. Würde man ihn einfach mit einem absoluten Wert vergleichen und danach entscheiden, ob das Bit „0“ oder „1“ ist, dann müßte man den Leser mit genau definierter Geschwindigkeit über die Strichcodezeilen führen. Um dies zu vermeiden, benötigt man einen Referenzwert, der von der Geschwindigkeit abhängig ist. Er wird beim Überfahren der beiden breiten Synchronstriche am Anfang einer Zeile gewonnen (siehe Listing nach START). Im Programm ist er in Speicherzelle REF1 (Referenz für „1“-Strich) abgelegt. Teilt man diesen Wert durch zwei (Befehl LSR), dann erhält man den Referenzwert für einen „0“-Strich REF0.

Die Routine AUSW vergleicht jeden gelesenen Strich mit REF0 und REF1. Ist der X-Register-Inhalt größer als der arithmetische Mittelwert, dann handelt es sich um eine „1“, ist er kleiner, handelt es sich um eine „0“. Das Programm setzt abhängig davon das Carry-Bit oder löscht es (zur späteren Weiterverwendung in BYTE).

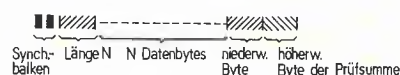
Wie ist der mc-Strichcode aufgebaut?

Jeder schwarze Strich stellt ein Bit dar: Ein breiter Strich bedeutet, daß dieses Bit „1“ ist, ein schmaler, daß es „0“ ist. Die Bits werden zu 8er-Gruppen (Bytes) zusammengefaßt. Links steht das niedrigstwertige Bit, rechts das höchstwertige. Das ASCII-Zeichen A (hexadezimal 41) sieht also folgendermaßen aus:



Ein schmaler Strich ist eine Einheit breit (ca. 0,3 mm); der Abstand zwischen zwei Strichen beträgt zwei Einheiten, und ein breiter Strich nimmt drei Einheiten ein. Jede Strichcodezeile beginnt mit

zwei breiten Synchronisationsbalken. Es folgt ein Byte, das die Anzahl der Datenbytes angibt. An die Datenbytes schließen sich niederwertiges und höherwertiges Byte einer 16-Bit-Prüfsumme an:



Die Prüfsumme wird durch Addition aller Datenbytes ermittelt. Grundsätzlich kann man auf diese Weise Programme in jeder Sprache veröffentlichen. Bei Basic-Programmen wird immer der volle ASCII-Text abgedruckt – also nicht der komprimierte Interncode (Tokens).

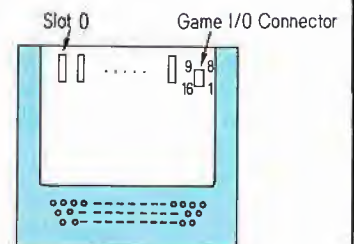
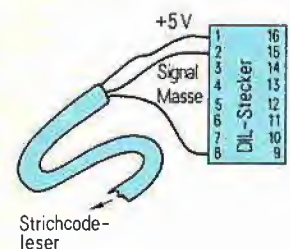


Bild 2. So wird der Strichcodeleser an den sogenannten „Game I/O Connector“ des Apple (rechts hinten, in der Nähe des Video-Ausgangs) angeschlossen. Das Programm ist so ausgelegt, daß Bit 7 eines Ports als Eingangsleitung dient

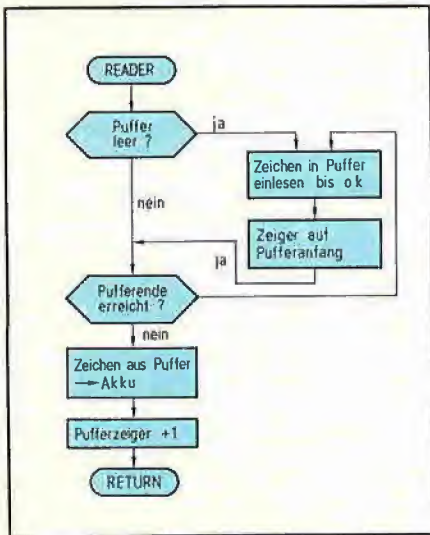
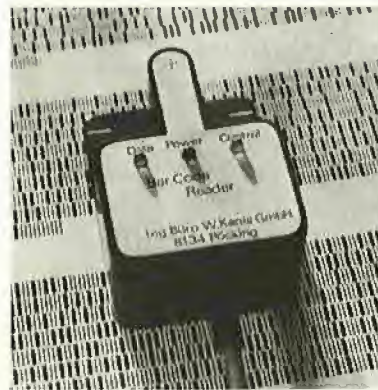


Bild 3. Das Programm READER übergibt dem aufrufenden Programm immer ein gültiges Zeichen im Akkumulator; es tritt beispielsweise an die Stelle der Tastaturabfrageroutine

Würden REF0 und REF1 während des gesamten Lesevorgangs gleich bleiben, dürfte sich auch die Geschwindigkeit nur unwesentlich ändern. Aus diesem Grund werden sie nach jedem gelesenen Bit aktualisiert. Das geht ganz einfach: War das Bit „0“, ist das Zählergebnis (X) der neue Referenzwert REF0 ($\text{REF0} \times 2$ ist REF1). War das Bit „1“, ist das Zählergebnis der neue Referenzwert REF1 ($\text{REF1}/2 = \text{REF0}$). Die zulässige Geschwindigkeitsvariation wird dadurch so groß, daß man sie praktisch nur überschreiten kann, wenn man mit dem Leser innerhalb der Zeile stehenbleibt.

Lesestift für mc-Programme



Programme mittleren Umfangs aus Zeitschriften abzutippen ist nicht nur zeitraubend, sondern auch fehlerintensiv. Ein Kassettenversand macht zwar die Tipparbeit überflüssig, aber er ist mit Wartezeit und zusätzlichen Kosten verbunden. mc erwog daher, künftig interessante Programme im Strichcode abzdrukken. Dieser Code kann schnell und fehlerlos in den Computer eingegeben werden. Schnell, weil man die Zeilen nur mit einem „Stift“ abzufahren braucht. Fehlerlos, weil eine Prüfsumme am Ende jeder Zeile dafür sorgt, daß nur richtig erkannte Zeichen übernommen werden. Der Teledaten-Service TEDAS hat diese Entwicklung allerdings überholt.

Weil Strichcodeleser bisher keine billige Angelegenheit waren, beauftragte mc eine Firma mit der Entwicklung eines geeigneten Produkts. Dieser Leser ist in mehrfacher Hinsicht außergewöhnlich.

Da er nur für flache Unterlagen gedacht ist, wurde er nicht als Stift, sondern als Element mit Auflagefläche ausgeführt (dieser Aufbau ist patentrechtlich geschützt).

Dadurch ist gewährleistet, daß die Optik immer senkrecht zum Papier steht. Mit einem relativ preiswerten Abtastelement läßt sich auf diese Weise eine genügend hohe Auflösung erzielen. Nur so konnte ein Preis realisiert werden, der wesentlich niedriger ist als der aller vergleichbaren Produkte auf dem Markt (135 DM inkl. MwSt. und Nachnahme-Versand bei: Ing.-Büro W. Kanis GmbH, Lindenberg 113, 8134 Pöcking). Der Leser wird mit 5 V versorgt und liefert ein Signal (schwarz/weiß) im TTL-Pegel. D. h., er kann an jeden Computer angeschlossen werden, der mindestens eine frei programmierbare Eingangsleitung hat. Ist kein sogenannter User-Port vorhanden, kann auch das Kassetten-Interface zweckentfremdet werden.

Ein Blick in Apple-DOS 3.3

Einen ausführlichen Bericht über DOS 3.3 haben wir bereits in mc 6/1983 veröffentlicht. Hier aber noch einige zusätzliche Informationen, die ebenfalls nützlich sind.

Die Tabelle nennt zu diesem Zweck die Zuordnung der logischen Sektoren auf der Diskette zu den Speicherbereichen, die DOS 3.3 in einem 48- oder 64-KByte-Apple belegt.

Hex-Adressen	Spur	Sektor
B600...B6FF	0	0
B700...B7FF	0	1
B800...B8FF	0	2
B900...B9FF	0	3

BA00...BAFF	0	4
BB00...BBFF*	0	5
BC00...BCFF*	0	6
BD00...BDFF*	0	7
BE00...BEFF*	0	8
BF00...BFFF*	0	9
nicht belegt	0	10, 11
9D00...9DFF	0	12
9E00...9EFF	0	13
9F00...9FFF	0	14
A000...A0FF	0	15
A100...A1FF	1	0
A200...A2FF	1	1
A300...A3FF	1	2
A400...A4FF	1	3
A500...A5FF	1	4

A600...A6FF	1	5
A700...A7FF	1	6
A800...A8FF	1	7
A900...A9FF	1	8
AA00...AAFF	1	9
AB00...ABFF	1	10
AC00...ACFF	1	11
AD00...ADFF	1	12
AE00...AEFF	1	13
AF00...AFFF	1	14
B000...B0FF	1	15
B100...B1FF	2	0
B200...B2FF	2	1
B300...B3FF	2	2

* Boot-Routine.

Edmund Möller

Apple-II steuert Fernschreiber

Gute ASCII-Drucker sind oft ebenso teuer wie ein kleiner Mikrocomputer; Baudot-Fernschreiber sind hingegen gebraucht recht günstig erhältlich und stellen trotz ihres eingeschränkten Zeichensatzes daher eine interessante Alternative zu Matrixdruckern dar. Der folgende Beitrag beschreibt die Ansteuerung eines solchen Fernschreibers mit dem Apple-II.

Nach einigen Ergänzungen und Versuchen ist aus [1] das hier beschriebene Programm entstanden. Um z. B. Programm-Listings besser aufbewahren zu können, wurde ein Zeilenzähler eingeführt, der nach jeweils 64 Zeilen 10 Leerzeilen einfügt, so daß das Endlospapier in DIN A4 große Abschnitte zerschnitten und bequem abgeheftet werden kann.

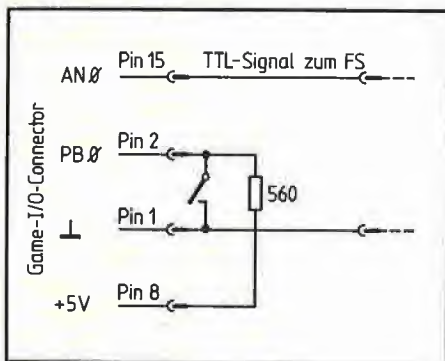


Bild 1. Hardware zum wahlweisen Zuschalten der Bildschirm-Ausgabe

Um den Drucker unabhängig vom Bildschirmausgabeprogramm betreiben zu können, wurde auch ein Zeichenzähler eingebaut, mit dessen Hilfe die volle Papierbreite von hier 64 Zeichen ausgenutzt werden kann. Ein Umschalter erlaubt es, das Bildschirmausgabeprogramm zuzuschalten. Dann werden maximal 40 Zeichen pro Zeile geschrieben. Dies ist bei schon existierenden Programmen manchmal nötig, weil der Zähler im Druckprogramm die ggf. vorhandenen TAB-Befehle nicht verarbeitet.

Einige Zeichen des ASCII-Zeichensatzes, die der Fernschreiber nicht besitzt, wurden durch vorhandene Zeichen ersetzt. Das Zeichen * für die Multiplikation wurde durch ein X ersetzt. Die Zeichen < und > werden durch runde Klammern dargestellt. Das ist sicherlich nicht ideal, aber man kann die richtige Bedeutung fast immer aus dem Zusammenhang erkennen. Statt des Semikolons (;) erscheint das Klingelzeichen, und die Anführungszeichen (") werden durch das Auslassungszeichen (') ersetzt.

In der hier beschriebenen Version belegt das Programm den Speicherbereich von \$7F00 bis \$7FFF und kann auf übliche Weise von der Kassette eingelesen werden. Dies ist für einen 32-KByte-Apple das obere Ende des RAM-Bereichs. Die Sicherung des Druckprogramms erfolgt in Basic durch Eingabe von HIMEM: 32510 im Direktmodus.

Das Programm wird aktiviert durch Belegen der Speicherplätze 36 und 37 in der Zero-Page mit der Anfangsadresse 7F00 oder in Basic durch POKE 54,0:POKE 55,127, was dasselbe bewirkt. Wenn das Programm später im EPROM liegt, kann der Aufruf über PR#... erfolgen. Die Rückkehr zur reinen Bildschirmausgabe erfolgt über PR#0.

Für die Ausgabe des seriellen Baudot-Signals und für das Festlegen des Ausgabemodus werden vorhandene Ports benutzt, die am Game I/O-Connector zur Verfügung stehen.

Die Ausgabe erfolgt über den Annunciator-Output AN0 (Game I/O-Connector Pin 15). Hier steht bereits TTL-Pegel zur Verfügung, mit dem direkt ein Transistor für die Steuerung des Fernschreiber-

Linienstroms angesteuert werden kann. Entsprechende Schaltungen wurden in ausreichender Zahl bereits veröffentlicht [2, 3].

Das Kriterium für den Ausgabemodus (Drucker allein oder mit dem Bildschirm) wird über einen Schalter dem Single-Bit-Input PB0 (Game I/O-Connector Pin 2) zugeführt (Bild 1).

Ein Listing des Programms ist in Bild 2 dargestellt. Hier hat sich das Programm selbst aufgelistet!

An zwei Stellen im Programm sind freie Speicherplätze vorhanden, die noch für kleine Ergänzungen verwendet werden können.

```

7F00- 20 4A FF A9 40 85 FE 85
7F08- FF A9 0A 85 FB A9 00 85
7F10- FA A9 1F 85 36 EA EA EA
7F18- EA EA EA EA EA D0 03 20
7F20- 4A FF A5 45 C9 8D 00 09
7F28- 20 6F 7F C6 FE F0 4F D0
7F30- 29 29 3F AA BD C0 7F 85
7F38- FD 29 20 C5 FC F0 0E 85
7F40- FC A8 F0 04 A9 1B D0 02
7F48- A9 1F 20 8F 7F A5 FD 20
7F50- 8F 7F E6 FA A5 FA C5 FF
7F58- B0 CE AD 61 C0 C9 80 B0
7F60- 0A EA EA EA EA 20 3F FF
7F68- 4C F0 FD 20 3F FF 60 A9
7F70- 08 20 8F 7F A9 02 20 8F
7F78- 7F A9 00 85 FA 60 20 6F
7F80- 7F C6 FB D0 F9 A9 40 85
7F88- FE A9 0A 85 FB D0 CB 48
7F90- AD 58 C0 20 BA 7F A0 04
7F98- 68 4A 90 07 48 AD 59 C0
7FA0- 4C A7 7F 48 AD 58 C0 20
7FA8- BA 7F 88 10 EB AD 59 C0
7FB0- 20 B5 7F 68 60 A9 35 20
7FB8- A8 FC A9 48 20 A8 FC 60
7FC0- 00 03 19 0E 09 01 0D 1A
7FC8- 14 06 0B 0F 12 1C 0C 18
7FD0- 16 17 0A 05 10 07 1E 13
7FD8- 1D 15 11 2D 00 00 00 00
7FE0- 24 34 25 29 3A 24 31 25
7FE8- 2F 32 1D 31 2C 23 3C 3D
7FF0- 36 37 33 21 2A 30 35 27
7FF8- 26 38 2E 2B 2F 3E 32 39
X
    
```

Bild 2. Hex-Dump des Fernschreiber-Ausgabeprogramms für einen 32-KByte-Apple. Es findet am oberen Speicherende Platz

Literatur

- [1] Baudot-Ausgabeprogramm für den 6502. FUNKSCHAU 1979, Heft 1.
- [2] Pietsch, H. J.: Amateurfunk-Fernschreibertechnik RTTY. RPB 25, Franzis-Verlag, München.
- [3] Fernschreiber-Ansteuerung. FUNKSCHAU 1979, Heft 26.

Ludwig Neidl

Apple-II sucht Bytes

Beim Aufsuchen von Strings in fremder Software gerät man mit den Möglichkeiten, die einem der Apple-II-Plus bietet, in arge Bedrängnis. Mit dem Monitorprogramm können nur Hexdumps eines vom Programmierer festgelegten Speicherbereiches erzeugt werden. Weiterhin ist es nicht möglich, den Speicher nach einer bestimmten Folge von Zeichen abzusuchen.

Wenn man wissen will, wie oft ein Unterprogramm in einem Maschinenprogramm aufgerufen wird, so helfen die Apple-Monitorbefehle auch nicht wei-

ter. Selbst das Absuchen des Speichers nach einem festgelegten Byte ist nicht möglich. Die Routine im Bild hilft hier aus dem

Dilemma. Sie besteht aus vier (fast) unabhängigen Teilen und wird über den User-Vektor (\$3F9, \$3FA) mit CTRL-Y sowie dem entsprechenden Codebuchstaben (W, F, S und Z) im Monitor aufgerufen. Das Programm wird in den Bereich \$2F5 bis \$3CD geschrieben und mit 2F5G initialisiert. Sollte jemand danach den gesamten Eingabepuffer (\$200 bis \$2FF) vollschreiben, so wird zwar der Initialisierungsteil zerstört, die Haupt-routinen bleiben jedoch erhalten. Stolz Besitzer einer Floppy können diese Hilfsprogramme ganz einfach mit BRUN ASCII/SEARCH (A\$2F5) initialisieren. Aufruf und Möglichkeiten jeder Routine seien hier im einzelnen erläutert:

Byte-Interpretation als ASCII-Zeichen

Der Aufruf erfolgt durch xxxx.yyyy CTRL-Y, wobei xxxx die Anfangs-adresse (hexadezimal) und yyyy die

**END OF PASS 1
**END OF PASS 2

```

0800      1  ; *****
0800      2  ; *
0800      3  ; *      ASCII / SEARCH
0800      4  ; *
0800      5  ; *      ROUTINE
0800      6  ; *
0800      7  ; *****
0800      8  ;
0800      9  ;
0800     10  CTYJMP EQU $3F8
0800     11  IN    EQU $200
0800     12  KBD   EQU $C000
0800     13  KBDSTB EQU $C010
0800     14  BELL  EQU $FF3A
0800     15  COUT  EQU $FDED
0800     16  CROUT EQU $FD8E
0800     17  GETLNZ EQU $FD67
0800     18  PRAI  EQU $FD92
0800     19  NXTAI EQU $FCBA
0800     20  KEYREA EQU $FD28
0800     21  MONZ  EQU $FF69
0800     22  ;
0800     23  ;
0800     24  YSAVE EPZ $34
0800     25  A1L   EPZ $3C
0800     26  A1H   EPZ A1L+1
0800     27  A2L   EPZ $3E
0800     28  A2H   EPZ A2L+1
0800     29  A4L   EPZ $42
0800     30  A4H   EPZ A4L+1
0800     31  YREG  EPZ $47
0800     32  ;
0800     33  ;
02F5     34      ORG $2F5
02F5     35      OBJ $800
02F5     36  ;
02F5     37  ;
02F5     38      LDA #START
02F7     39      STA CTYJMP+1
02FA     40      LDA /START
02FC     41      STA CTYJMP+2
02FF     42      RTS
0300     43  ;
0300     44  ;
0300     45  START LDY YSAVE
0302     46      INC YSAVE
0304     47      LDA IN,Y
0307     48      CMP #"W"
0309     49      BEQ ASC
030B     50      CMP #"F"

```

```

030D F032      51      BEQ ASCSRC
030F C9D3      52      CMP #"S"
0311 F078      53      BEQ SEARCH
0313 C9DA      54      CMP #"Z"
0315 F070      55      BEQ ZSRCH
0317 4C3AFF     56      JMP BELL
031A           57  ;
031A           58  ;
031A A53C      59      MOD32C LDA A1L
031C 291F      60      AND #$1F
031E D006      61      BNE DATOUT
0320 208EFD     62      ASC   JSR CROUT
0323 20AB03     63      JSR ADWRRI
0326 B13C      64      DATOUT LDA (A1L),Y
0328 100A      65      BPL DOIT
032A C9A0      66      CMP #" "
032C 900F      67      BCC CTRL
032E C9E0      68      CMP #$E0
0330 9002      69      BCC DOIT
0332 E920      70      SBC #$20
0334 20EDFD     71      DOIT  JSR COUT
0337 20BAFC     72      JSR NXTAI
033A 90DE      73      BCC MOD32C
033C 60        74      RTS
033D A92E      75      CTRL  LDA #$2E
033F D0F3      76      BNE DOIT
0341           77  ;
0341           78  ;
0341 2067FD     79      ASCSRC JSR GETLNZ
0344 8634      80      STX YSAVE
0346 38        81      SEC
0347 A53E      82      LDA A2L
0349 CA        83      DEX
034A 863E      84      STX A2L
034C E53E      85      SBC A2L
034E 853E      86      STA A2L
0350 B002      87      BCS NEWONE
0352 C63F      88      DEC A2H
0354           89  ;
0354 A000      90      NEWONE LDY #00
0356 B13C      91      LOOPAS LDA (A1L),Y
0358 1006      92      BPL PASCII
035A C9E0      93      CMP #$E0
035C 900A      94      BCC COMPAR
035E E920      95      SBC #$20
0360 09C0      96      PASCII ORA #%11000000
0362 C9E0      97      CMP #$E0
0364 9002      98      BCC COMPAR
0366 E940      99      SBC #$40
0368           100 ;
0368 D90002    101      COMPAR CMP IN,Y
036B D014      102      BNE NXTONE
036D C8        103      INY

```

Die Interpretation von Bytes als ASCII-Zeichen sowie das Suchen nach ASCII-Zeichen, Bytes und Doppelbytes gestattet dieses Programm für den Apple-II-Plus


```

036E C434 104 CPY YSAVE 03A2 A447 133 LDY YREG
0370 90E4 105 BCC LOOPAS 03A4 C8 134 ZAEHL INY
0372 20AB03 106 JSR ADWRRI 03A5 20BAFC 135 NEXTON JSR NXTA1
0375 A000 107 LDY #00 03A8 90E3 136 BCC ZWEIG
0377 B13C 108 FOLLOOP LDA (A1L),Y 03AA 60 137 RTS
0379 20EDFD 109 JSR COUT 03AB 138 ;
037C C8 110 INY 03AB 139 ;
037D C434 111 CPY YSAVE 03AB 2C00C0 140 ADWRRI BIT KBD
037F 90F6 112 BCC FOLLOOP 03AE 1016 141 BPL RETAD2
0381 20BAFC 113 NXTONE JSR NXTA1 03B0 2028FD 142 JSR KEYREA
0384 90CE 114 BCC NEWONE 03B3 C9A0 143 CMP # " "
0386 60 115 RTS 03B5 D008 144 BNE CTRLC
0387 116 ; 03B7 2C00C0 145 SPACE BIT KBD
0387 A000 117 ZSRCH LDY #00 03BA 10FB 146 BPL SPACE
0389 F002 118 BEQ ZWEIG 03BC 2078FD 147 JSR KEYREA
038B 119 ; 03BF C983 148 CTRLC CMP #83
038B 120 ; 03C1 D003 149 BNE RETAD2
038B A001 121 SEARCH LDY #01 03C3 68 150 PLA
038D 122 ; 03C4 68 151 PLA
038D B13C 123 ZWEIG LDA (A1L),Y 03C5 60 152 RTS
038F D94200 124 CMP A4L,Y 03C6 2092FD 153 RETAD2 JSR PRA1
0392 D011 125 BNE NEXTON 03C9 A9A0 154 LDA # " "
0394 88 126 DEY 03CB 4CEDFD 155 JMP COUT
0395 3006 127 BMI ZWRI 03CE 156 ;
0397 B13C 128 LDA (A1L),Y 03CE 157 ;
0399 C542 129 CMP A4L 158 END
039B D007 130 BNE ZAEHL
039D 8447 131 ZWRI STY YREG
039F 20AB03 132 JSR ADWRRI
***** END OF ASSEMBLY

```

aabb. Intern steht eine Adresse in der Reihenfolge Lower Byte, Higher Byte (also bb-aa) im Speicher. Durch den Aufruf erfolgt diese Vertauschung automatisch.

Suche nach beliebigen Bytes

Dafür muß man eingeben: aa<xxxx.yyyy CTRL Y Z. Die Routine sucht dann nach dem spezifizierten Byte im angegebenen Speicherbereich.

Der ASCII-Dump und alle Suchroutinen können durch das Drücken von Space oder CTRL C unterbrochen bzw. beendet werden. Will man nach Betätigung der Space-Taste weitermachen, so drückt man wieder auf Space. Will man die Suche oder den Dump ganz beenden, so drückt man CTRL C.

Das Listing im Bild wurde mit Hilfe des LISA-Assemblers erstellt und der Quellcode direkt beim Assemblieren von einer Olympia-ES100-Schreibmaschine gedruckt.

Endadresse darstellt. Das W muß direkt hinter das CTRL Y geschrieben werden. Nach dem Aufruf erfolgt ein ASCII-Dump des angegebenen Bereichs. Da beim Apple-II die Zeichen auch invers (\$00 bis \$3F) und im Flashing-Modus (\$40-\$7F) dargestellt werden können, berücksichtigt die Routine den gesamten Darstellungsbereich von \$00 bis \$FF. Hexadezimale Zahlen zwischen \$80 und \$9F (Control-Zeichen) werden als inverser Punkt dargestellt. Da man auch Programme mit Kleinbuchstaben (ab \$E0) untersuchen könnte, den Dump auch interpretieren will, erfolgt für den Bereich \$E0 bis \$FF eine Konversion in Großbuchstaben.

Ein ähnliches Programm ist dem Autor schon bekannt [1], dieses ist jedoch nicht für den Apple-II-Plus geeignet, da es auf Sweet-16-Routinen zurückgreift, die nur im „alten“ Betriebssystem existieren.

Suche nach ASCII-Zeichen

Hier erfolgt der Aufruf durch xxxx.yyyy CTRL Y F. Die Bedingungen für die einzelnen Parameter sind die gleichen wie beim ASCII-Dump. Nach dem Aufruf springt der Cursor eine Zeile tiefer, und man kann eine Zeichenkette mit bis zu 255 Zeichen eingeben. Die Routine sucht nach diesem String im angegebenen Bereich, ob der String nun normal, invers, blinkend oder in Kleinbuchstaben im Speicher steht, und druckt dann die Startadresse des Strings (hexadezimal) sowie den String in seiner ursprünglichen Form. Kleinbuchstaben werden dabei natürlich nicht richtig dargestellt.

Suche nach Adressen

Der Aufruf geschieht mit aabb<xxxxx.yyyy CTRL Y S. Die Routine sucht im Bereich xxxx bis yyyy nach der Adresse

Literatur

[1] Hex-ASCII Memory Dump, The Apple Orchard, March/April 1980, Page 79.

Logikanalysator als Apple-Zusatzseinheit

Das jüngste und bisher anspruchsvollste Produkt der Firma Northwest Instrument Systems (Hauptstr. 17, 8180 Tegernsee) ist ein Logikanalysator, der als Bedienungs- und Steuereinheit einen Apple-II benötigt. Im Apple selbst steckt eine Interfacekarte, von der ein Bandkabel zum separaten Gehäuse (µAnalyst 2000) führt. Dort befinden sich die Zeitsteuerlogik und bis zu fünf 16-Kanal-Karten mit Speicher und Tastkopf. Außerdem ist im Zusatzgehäuse eine eigene Stromversorgung untergebracht. Nach Angaben des Unternehmens erlaubt es das Konzept, Logikanalysatoren herkömmlicher Art um den Faktor 2...5 zu verbilligen. Der US-Preis für das Grundgerät (ohne Apple) mit 1 KByte Speicher liegt derzeit bei 2500 \$. 16-Kanal-Einschübe mit 1 KByte Speicher kosten knapp 700 \$, für die 4-KByte-Einschübe muß man 200 \$ mehr ausgeben. Das Prädikat „low cost“ will Northwest Instrument Systems aber nicht als Qualitätsmerkmal verstanden wissen. Europa-Vertriebschef Ron Imbriale verweist darauf, daß im neuen Gerät Möglichkeiten vorhanden sind, die man als einzigartig bezeichnen kann. So ist es in einer zum

Patent angemeldeten Betriebsart (Multiple Preview Acquisition) möglich, alle Stellen eines Programms zu lokalisieren, von denen aus ein bestimmtes Unterprogramm aufgerufen wird. In einer anderen Betriebsart lassen sich Signale in Abhängigkeit von den erfaßten Daten erzeugen. Besonders nützlich ist das, wenn man Interrupts auslösen will, um beispielsweise die Registerinhalte auszugeben. Eine weitere Besonderheit des Gerätes ist die durchgehende symbolische Darstellung von Zuständen. Der Benutzer arbeitet mit Namen, so wie er es von der µC-Programmierung her gewohnt ist.

Zum Betrieb der Software, die in Pascal geschrieben ist, reicht ein Grundgerät mit Disketten-Betriebssystem aus. Da das zugehörige Programmpaket Standarddatenformate benutzt und vollständig dokumentiert ist, hat der Anwender die Möglichkeit, die erfaßten Daten so weiterzuverarbeiten, wie es für seinen Anwendungsfall erforderlich ist. Beispiele dafür sind Datenreduktion, Leistungsmessungen und grafische Darstellungen. Ho.

Jürgen Müller

Autostart und Programm-schutz für Apple-II-plus

Die Möglichkeit, Programme so zu laden, daß sie sich selbst starten, ist gerade bei der Benutzung eines Kassettenrecorders als Speichergerät oft wünschenswert. Mit einer solchen Einrichtung können Programme beispielsweise Variablenfelder oder Maschinen-Unterprogramme vom Band übernehmen, ohne daß vom Benutzer weitere Operationen auszuführen sind. Besonders die Möglichkeit einer recht komfortablen „Dateiverwaltung“ in Verbindung mit den Befehlen STORE und RECALL ist sehr interessant.

Während zur Realisierung des Autostarts für einige Mikrocomputer Programme ausgearbeitet wurden, die sich teilweise ziemlich trickreicher Techniken bedienen, ist er beim Apple II plus mit minimalem Aufwand möglich. Die Autostart-Einrichtung wurde nämlich im Basic-Interpreter bereits eingebaut! Die Handbücher schweigen sich über diese interessante Möglichkeit aller-

dings aus – wohl, weil der Autostart ursprünglich als Programmschutz konzipiert wurde, der verständlicherweise nicht allgemein bekannt werden sollte.

Wie funktioniert also der Autostart, und was bewirkt er? Die, wohl jedem Apple-Benutzer bekannte, kurze Unterbrechung des Kenntons vor den Basic-Programmaufzeichnungen enthält eine In-

formation von drei Bytes, die zum Laden des eigentlichen Programmes benötigt wird: Zwei Bytes geben die Länge des Programmes an – das dritte aber stellt eine Art „Schutz-Flag“ dar. Es wird zunächst in den Speicher \$D6 kopiert (Bild 1). Nachdem dann das Basic-Programm geladen wurde, wird \$D6 abgefragt. Ist Bit 7 gesetzt, so erfolgt nicht – wie üblich – ein Warmstart des Basic-Interpreters, sondern es wird sofort mit der Ausführung des Programms begonnen.

Um ein Programm selbststartend abzuspeichern, ist das Schutz-Flag vorher zu ändern. Die Befehlsfolge lautet:

POKE 82, 247 : SAVE : POKE 82, 85
Es tritt allerdings noch eine weitere Besonderheit auf: Ein per Autostart geladenes Programm kann zwar mit Reset, ctrl C oder mit einem END-Befehl im Programm unterbrochen werden; aber jedes daraufhin eingegebene Statement wird ignoriert, und der Interpreter beginnt erneut mit der Programmausführung. Ein beinahe perfekter Programmschutz also – ist das Programm erst einmal geladen, so können weder Listings noch Kopien hergestellt werden, und auch Änderungen sind unmöglich. Falls dieser Effekt unerwünscht ist, muß der Schutz vom Programm aus deaktiviert werden. Dies ist mit dem Befehl POKE 214,0 ohne weiteres möglich. Er kann gleich am Anfang des Programmes eingesetzt werden; denkbar wäre auch, daß bei einer Eingabe überprüft wird, ob der Benutzer ein spezielles Schlüsselwort gewählt hat, das dann die Aufhebung des Schutzes und einen Programmstopp bewirkt.

Lädt man ein Programm, das selbststartend abgespeichert wurde, so setzt der Interpreter den Pointer auf das Programmende nicht. Dies beeinträchtigt zwar die Lauffähigkeit des Programms nicht, führt jedoch bei nachträglichen Änderungsversuchen oder SAVE-Operationen zum Verlust des Programms.

Als einfache Abhilfe sollte in Programmen, bei denen nicht sowieso ein Schutz erwünscht ist, zu Beginn die Zeile POKE 175, PEEK (105) : POKE 176, PEEK (106) eingeführt werden.

Aber wie bereits erwähnt: Der mit dem Autostart erzielte Programmschutz ist leider nur fast perfekt. Soll ein „geschütztes“ Programm doch noch geändert werden, obwohl es keine Schutz-

d8c9-	20 f0 d8	jsr	d8f0	Bereichsgrenzen setzen
d8cc-	20 fd fe	jsr	fe fd	Bereich vom Band lesen
d8cf-	18	clc		
d8d0-	a5 67	lda	67	Programmanfang L
d8d2-	65 50	adc	50	+ Programmlänge L
d8d4-	85 69	sta	69	= Programmende L
d8d6-	a5 68	lda	68	Programmanfang H
d8d8-	65 51	adc	51	+ Programmlänge H
d8da-	85 6a	sta	6a	= Programmende H
d8dc-	a5 52	lda	52	
d8de-	85 d6	sta	d6	Schutzflag in D6 speichern
d8e0-	20 01 d9	jsr	d9 01	Bereichsgrenzen (Basic-Progr.)
d8e3-	20 fd fe	jsr	fe fd	Basic-Programm einlesen
d8e6-	24 d6	bit	d6	Schutzflag gesetzt?
d8e8-	10 03	bpl	d8ed	
d8ea-	4c 65 d6	jmp	d6 65	ja
d8ed-	4c f2 d4	jmp	d4 f2	nein, normal weiter

Bild 1. Die LOAD-Routine des Basic-Interpreters: Aus den im Vorspann gelesenen Informationen wird die Endadresse des Programmes errechnet; dann wird das Schutzflag nach D6 kopiert. Ersetzt man den LDA-Befehl (D8DC...D8DF) in einer RAM-Kopie der Routine durch NOP, dann kann man den Schutz umgehen

0300-	a0 f0	ldy	+f0	Output-Adresse L
0302-	a9 fd	lda	+fd	Output-Adresse H
0304-	84 36	sty	36	Normalwert einsetzen
0306-	85 37	sta	37	
0308-	4c ad 00	jmp	00ad	Sprung ins User-Programm

Bild 2. So einfach kann ein Autostart-Programm für Maschinensprache aussehen (wer ganz korrekt programmieren will, baut noch zwei PLAs ein): Die Sprungadresse ist natürlich durch die Startadresse des jeweiligen User-Programmes zu ersetzen. Abspeicherung mit 36:0 3 N 30.XXXXW 36:F0 FD

Achtung: Die Kontrolle von Lesefehlern wird durch diese Routine außer Kraft gesetzt!

Abschaltung enthält, so gibt es recht einfache Methoden, den Schutz zu knacken. Wenn man vor dem Laden des zu „überlistenden“ Programmes den Soft-Entry-Vector ändert, so daß bei Drücken der Reset-Taste der Monitor gestartet wird, ist es nicht schwierig, den Schutz abzuschalten. Es ist einzugeben:

POKE 1010, 105 : POKE 1011, 255 :
CALL-1169 : LOAD

Wenn das Laden beendet ist und das Programm läuft, kann Reset gedrückt werden. Im Monitor gibt man ein:

und das Programm kann wie gewohnt bearbeitet werden. Eine andere Möglichkeit wäre, die LOAD-Routine in den RAM-Bereich zu verschieben und so zu ändern, daß das Schutz-Flag nicht mehr abgefragt wird.

Abschließend ein Tip für Maschinensprache-Programmierer und Besitzer anderer Apple-II-Typen. Ein Autostart von Maschinenprogrammen, der auf allen Apples funktioniert, ist über die Änderung des CSW-Vektors (\$36,37) möglich. Diese Speicherplätze geben die Adresse der Output-Routine an. Sie können auf eine Autostart-Routine gesetzt werden, die zunächst den Vektor wieder auf den normalen Wert setzt und dann ins User-Programm springt (Bild 2). Die Abspeicherung ist vorzunehmen mit 36:XX XX N 30.: EEEE
W 36 : F0 FD
XX XX ist die Anfangsadresse der Autostart-Routine, EEEE die Endadresse des Hauptprogramms. Der N-Befehl dient nur zur Trennung der beiden Kommandos, die unbedingt in der angegebenen Form direkt aufeinanderfolgend einzugeben sind.

3. Ferner muß noch folgendes eingegeben (bzw. geändert) werden:
\$8350:F4 / \$8415:F4 / \$8418:F3
\$8EC2:C6 34 20 75 FE 4C 43 F3
\$8F6A:A3 / \$8FCC:B2 / \$8FCF:ED
\$8FD2:EC / \$8FE3:C9 / \$8FE9:C3
\$8FFC:59 FF

Hat man alles eingetippt, gebe man noch das im Bild wiedergegebene Verschiebungs- und Initialisierungsprogramm ein und speichere das Ganze mit

BSAVE Name,A\$8300,L\$D00.

Mit BRUN Name (bzw. 8300G) kommt man sofort in den erweiterten „Monitor“ und MiniAssembler“ (F666G). Mit C081 aktiviert man den normalen (ROM-)Monitor, und mit C080 kommt man wieder zurück.

Damit man weiß, wo man sich befindet, ist das Bereitschaftszeichen des erweiterten Monitors geändert worden: „#“ statt „*“ (\$8F6A:A3 statt AA). Damit man sich nicht versehentlich in der Karte „aufhängt“, sind folgende Maßnahmen getroffen worden:

1. „CTRL C“ ist im erweiterten Monitor außer Betrieb gesetzt.
2. „Reset“ führt in den erweiterten Monitor zurück (Achtung: Falls Reset benutzt wurde, ist nach Rückkehr in den ROM-Monitor Reset zu wiederholen, bevor man DOS benutzen kann).
3. „INT“ führt von Basic aus nicht in die Karte (siehe Zeile 8323 im Maschinenprogramm).

Friedhelm Hellmann

Step und Trace für Apple-II+ und Apple-IIe

Für Besitzer der 16-KByte-Speichererweiterung besteht eine wenig aufwendige Möglichkeit, sich die Step- und Trace-Befehle des alten Monitors zur Verfügung zu stellen.

Man verwendet das von der Master-Diskette ladbare Integer-Basic, das eine Kopie des Autostart-Monitors enthält. Es ist zweckmäßig, nur den Teil davon zu behalten, der den MiniAssembler enthält (etwa ab Adresse \$F500). Im freiwerden Teil der Karte ist dann Platz für eine Erweiterung des Monitors. Eine Erweiterung hat den Vorteil, daß die Verbesserungen des Autostart-Monitors (z. B. die ESC-Befehle) nicht verlorengehen. Alles zusammen (MiniAssembler und erwei-

terter Monitor) paßt in den Bereich \$F340 bis \$FFFF. Man lädt das Ganze zunächst in den Bereich \$8340 bis \$8FFF und verschiebt es dann mit einem gesonderten Programm in die Karte.

Im einzelnen gehe man so vor:

1. Man lade INTBASIC mit BLOAD INT-BASIC,A\$6000.
2. Folgende Bereiche müssen aus dem (alten) Monitorlisting des Apple-Handbuchs in die angegebenen Speicherplätze eingegeben werden:
\$FA40...\$FAD6 → \$8340...\$83D6
\$FAFD...\$FB1D → \$83FD...\$841D
\$FB60...\$FBC0 → \$8460...\$84C0

8300-	A0 00	LDY	#00
8302-	A9 83	LDA	#83
8304-	84 E0	STY	#E0
8306-	85 E1	STA	#E1
8308-	A9 F3	LDA	#F3
830A-	84 E2	STY	#E2
830C-	85 E3	STA	#E3
830E-	A0 40	LDY	#40
8310-	2C 81 C0	BIT	#C081
8313-	2C 81 C0	BIT	#C081
8316-	B1 E0	LDA	(#E0),Y
8318-	91 E2	STA	(#E2),Y
831A-	C8	INY	
831B-	D0 F9	BNE	#8316
831D-	E6 E1	INC	#E1
831F-	E6 E3	INC	#E3
8321-	D0 F3	BNE	#8316
8323-	8C 00 E0	STY	#E000
8326-	2C 80 C0	BIT	#C080
8329-	4C 69 FF	JMP	#FF69

Dieses Programm verschiebt die eingetippten Teile des alten Monitors und startet den erweiterten Monitor

Hans-Georg Joepgen

DOS-Umschaltung beim Apple

Das Disketten-Betriebssystem DOS 3.3 für Computer der Apple-Klasse enthält zwar Dienstleistungen zur Konversion von Aufzeichnungen ins neue 16-Sektor-Format und Handreichungen zur Rückschaltung auf 13-Sektor-Betrieb, jedoch ist simultanes Bearbeiten von Disketten in beiden Aufzeichnungsformaten nicht ohne weiteres möglich. Abhilfe bringt das hier vorgestellte Programm „DDS“ („Double DOS Switch“), das darüber hinaus – mit gewissen Einschränkungen freilich – Nutzung des neuen Disketten-Betriebssystems ohne die sonst erforderlichen Hardware-Modifikationen (Austauschen von ROMs) erlaubt.

Seit dem Erscheinen von DOS 3.2 Mitte 1979 sind zahlreiche Hardware-Ergänzungskarten (wie beispielsweise „Doublevision“) und Software-Pakete (hierzu zählen „APEX“ und „CP/A“) entwickelt worden, die nicht aus den Häusern Apple oder ITT selbst stammen. Im Unterschied zu Original-Zubehör treten bei diesen von Fremdherstellern produzierten Erzeugnissen durchaus Inkompatibilitäten zu 16-Sektor-Disketten auf. Will man auf die überzeugenden Vorzüge von DOS 3.3 nicht verzichten und dennoch voll auf die für die ältere DOS-Version entwickelte Hard- und Software ungeschmälert zurückgreifen, dann ist es wünschenswert, beide Systeme wechselseitig zur Verfügung zu haben. Neubooten brächte keinen Ausweg, weil dabei ein beträchtlicher Speicherbereich überschrieben wird – hier setzt „DDS“ an.

Beide DOS-Versionen residieren im Speicherbereich hex 9600...BFFF, wobei die ersten Seiten Buffer für drei Files darstellen (Maxfiles = 3). Darüber hinaus bedienen sich beide Betriebssysteme jeweils einiger Adressen auf den Seiten 0 und 3 (0000...00FF und 0300...03FF), wozu noch einige Bytes aus Lücken des Bildschirm-Speichers kommen, die beim Umschalten jedoch außer Betracht bleiben können. Da keine präziseren Veröffentlichungen über Art und Funktion des innerhalb der Seiten 0 und 3 in Anspruch genommenen Schreib-Lese-Speichers vorlagen, fiel die Entschei-

dung zugunsten einer Art „Holzhacker-Methode“: Beim Übergang von einem DOS zum anderen werden außer DOS und seinen drei File-Buffern die Seiten 0 und 3 jeweils komplett aus den DDS-Buffern nachgeladen: Nicht sehr elegant und mit überreichlich Redundanz wohl-versehen, aber: Es funktioniert!

Monitor-„MOVE“ ungeeignet

Bei ersten Versuchen wurden, während der Rechner unter DOS 3.3 lief, zuvor unter DOS 3.2 aufgenommene Abbilder der drei Blöcke Seite 0, Seite 3 und schließlich der Bereich aufwärts von 9600 per Kassette eingespielt. Dabei zeigte sich schnell, daß dies so mit der Seite 0 nicht funktionierte. Denn: Bei Schreib-Lese-Operationen bedient sich der Monitor selbst unmittelbar der Seite 0 und kommt so außer Tritt. Weiterhin erwies es sich aus dem gleichen Grunde als unmöglich, die vom Apple-Betriebssystem angebotene Monitor-Routine „MOVE“ (Kopieren eines Adreßbereichs in einen anderen) mitzubeneutzen. Es war deswegen erforderlich, einen eigenen „MOVER“ zu schreiben, wobei die Wahl auf eine Routine mit Parameter-Übergabe per Hauptprogramm-Liste fiel. Das Assembler-Listing (Bild) ist weitgehend selbsterläuternd ausgelegt, so daß nur wenige zusätzliche Bemerkungen erforderlich sind. Bei der Definition der Adressen-Bereiche bedeutet der Buchstabe Z (Zero) die Zuordnung zur Seite 0, T (wie Three) zur Seite 3 und der Buchstabe D schließlich besagt Zugehö-

rigkeit zum DOS-Hauptbereich. Die Labels ZBUF, TBUF und DBUF bezeichnen den jeweiligen Beginn des Bufferbereiches, von dem aus durch MOVER das betreffende DOS an seinen Arbeitsort kopiert wird. Nachgesetzte Suffixe sagen, welche DOS-Version hier abgespeichert ist: 32 steht für DOS 3.2, die Ziffern 33 für das 16-Sektor-Format DOS 3.3.

Besondere Erwähnung verdient die Subroutine ALI: Sie stattet unsere ITT-2020 oder unseren Apple-II nämlich mit einer ungemein nützlichen Adressierungsart aus, wie sie der CPU 6502 nicht unmittelbar zur Verfügung steht: Der Akkumulator wird mit dem Inhalt jener Speicherzelle geladen, deren Seitenbezeichnung im Y-Register und deren Einzeladresse im X-Register steht. Vor diesem „vollindiziert-indirektem“ Ladevorgang erfolgt eine auf den durch die beiden Indexregister gebildeten Doppelzeiger bezogene Auto-Inkrement-Operation. Für ALI bieten sich auch außerhalb von Double DOS Switch und MOVER allerlei interessante Anwendungsmöglichkeiten; nur möge man sich davor hüten, ALI in EPROMs zu brennen: Die Subroutine ist selbstmodifizierend und somit nicht „ROMabel“, bietet dafür jedoch den Vorteil, ohne Rückgriff auf die Seite 0 auszukommen.

Installation und Bedienung

Die Erstinstallation von Double DOS Switch fordert einige Arbeit, führt dann jedoch zu einer Diskette, die ein Folgeladen fortan dafür um so bequemer macht: Man ruft den Monitor auf und stellt sich erst einmal Bandaufzeichnungen der jeweils drei Blöcke eines jeden Disk Operating Systems her. Hierbei ist es erforderlich, die beiden Nullseiten-Blöcke vor dem Abspeichern auf Kassette mit Hilfe des M-Monitor-Befehles erst einmal auszulagern: Empfohlen wird der Bereich hex 1000 bis hex 10FF; von dort aus kann ohne Schwierigkeiten aufgezeichnet werden. Sodann gibt man die 175 Bytes aus dem Assembler-Listing ein, verbessert die unvermeidlichen

Tipp-Fehler und spielt zuletzt die sechs
 Bandaufzeichnungen zurück in den
 Rechner: Und zwar an die entsprechen-
 den BUF-Positionen, die man der „Sym-
 bol Table“ des Assembler-Listings ent-
 nimmt. Wir sind noch immer im Moni-
 tor und speichern jetzt unser Werk
 durch die Anweisung „BSAVE DOUBLE
 DOS SWITCH, A\$3C00,L22702“ auf
 Diskette. Sind uns bis jetzt keine unkor-
 rigierten Fehler verblieben, genügt fort-
 an die schlichte Anweisung „BLOAD
 DOUBLE DOS SWITCH“ zur Folge-In-
 stallation des Programms.

Wie nun arbeitet man mit diesem „DOS-Doppelschalter“? Geschaffen wurde er hauptsächlich als Subroutine, der sich ihrerseits wieder andere in Maschinensprache formulierte Programme bedienen, die dann natürlich alle unterhalb der Adresse hex 3C00 angesiedelt sein müssen. Um DOS 3.2 zum Leben zu erwecken, bedarf es nur noch eines simplen „JSR 3C00“. Möchte man im neuen 16-Sektor-Format weiterarbeiten, ruft man Double DOS Switch durch „JSR 3C1C“. Vom Monitor aus benutze man zweckmäßigerweise den „G“-Befehl. Wiewohl für den Zugriff von Maschinensprache-Programmen aus geschrieben, stehen die DDS-Dienstleistungen auch von Basic aus zur Verfügung. Man muß hier freilich durch Zurücksetzen des HIMEM-Pointers dafür Sorge tragen, daß sich Basic und Double DOS Switch nicht wechselseitig auf die Füße treten. Initialisierung von DOS 3.2 erfolgt dann durch „CALL 15360“, DOS 3.3 wird durch „CALL 15388“ in Aktion gesetzt. Nach einer solchen Umschalt-Prozedur tut man gut daran, durch die Kommandos „FP“ und „INT“ die neu installierte DOS-Version zum Trittassen zu bewegen; unter gewissen (noch nicht völlig aufgeklärten) Randbedingungen können sonst Betriebsanomalien auftreten. Voraussetzung dieser Korrekturprozedur: Beide Standard-Basic-Dialekte (also Integer-Basic und Applesoft oder Palsoft) müssen resident zur Verfügung stehen.

Betriebserfahrungen

Entwickelt und erprobt wurde Double DOS Switch auf einem Computer ITT 2020 mit Palsoft im ROM unter DOS 3.3 – installiert war in Slot 0 eine Integer-Karte mit zusätzlichem Single-Step-Monitor, während der Rechner selbst das Autostart-ROM enthielt. In dieser Konfiguration arbeitete das Programm einwandfrei; bei Systemwechsel von Basic aus quitierte der Rechner die entsprechenden CALLs gelegentlich mit einem

```

ZSTART EQU 0
ZEND EQU 255
TSTART EQU $300
TEND EQU $3FF
DSTART EQU $9600
DEND EQU $BFFF
ZLEN EQU ZEND-ZSTART
TLEN EQU TEND-TSTART
DLEN EQU DEND-DSTART
*
* ORG $3C00
*
* GET DOS 3.2
* -----
3C00-20383C JSR MOVER
3C03-AE3C DFD ZBUF32
3C05-AD3D DFD ZEND32
3C07-0000 DFD ZSTART
3C09-20383C JSR MOVER
3C0C-AE3D DFD TBUF32
3C0E-AD3E DFD TEND32
3C10-0003 DFD TSTART
3C12-20383C JSR MOVER
3C15-AE3E DFD DBUF32
3C17-AD68 DFD DEND32
3C19-0096 DFD DSTART
3C1B-60 RTS
*
* GET DOS 3.3
* -----
3C1C-20383C JSR MOVER
3C1F-AE68 DFD ZBUF33
3C21-AD69 DFD ZEND33
3C23-0000 DFD ZSTART
3C25-20383C JSR MOVER
3C28-AE69 DFD TBUF33
3C2A-AD6A DFD TEND33
3C2C-0003 DFD TSTART
3C2E-20383C JSR MOVER
3C31-AE6A DFD DBUF33
3C33-AD94 DFD DEND33
3C35-0096 DFD DSTART
3C37-60 RTS
*
* SUBROUTINE MOVER
* -----
3C38-68 MOVER PLA
3C39-AA TAX
3C3A-68 PLA
3C3B-A8 TAY
3C3C-206F3C JSR ALI
3C3F-8D7E3C STA FROML
3C42-206F3C JSR ALI
3C45-BD7F3C STA FROMH
3C48-206F3C JSR ALI
3C4B-8DAD3C STA TILL
3C4E-206F3C JSR ALI
3C51-8DAD3C STA TILH
3C54-206F3C JSR ALI
3C57-8D813C STA TOL
3C5A-206F3C JSR ALI
3C5D-8D823C STA TOH
3C60-206F3C JSR ALI
3C63-8E6D3C STX RETAD
3C66-8C6E3C STY RETAD+1
3C69-207D3C JSR MOVE
3C6C-4C DFB $4C JMP
3C6D-0000 DFD 0 RETURN ADD
*
* INC XY, LDA (XY);
*
3C6F-E8 ALI INX
3C70-D001 BNE TFER
3C72-C8 INY
3C73-8E7A3C TFER STX LLOAD
3C76-8C7B3C STY LHIAD
3C79-AD DFB $AD LDA ABS
3C7A-EA LLOAD NOP LO ADDR
3C7B-EA LHIAD NOP HI ADDR
3C7C-60 RTS
*
* MOVE FROM/TIL/TO
*
3C7D-AD MOVE DFB $AD LDA
3C7E-EA FROML NOP
3C7F-EA FROMH NOP
3C80-8D DFB $8D STA
3C81-EA TOL NOP
3C82-EA TOH NOP
3C83-20893C JSR INCCMP
3C86-B0F5 BCS MOVE
3C88-60 RTS
*
* INC AND COMPARE;
*
3C89-EE7E3C INCCMP INC FROML
3C8C-D003 BNE ICNT01
3C8E-EE7F3C INC FROMH
3C91-EE813C ICNT01 INC TOL
3C94-D003 BNE ICNT02
3C96-EE823C INC TOH
3C99-AD7F3C ICNT02 LDA FROMH
3C9C-CDAD3C CMP TILH
3C9F-9009 BCC IEND1
3CA1-ADAC3C LDA TILL
3CA4-CD7E3C CMP FROML
3CA7-B002 BCS IEND2
3CA9-60 RTS C=0; DONE
3CAA-38 IEND1 SEC
3CAB-60 IEND2 RTS C=1; GO ON
*
* LOCAL BUFFERS;
*
3CAC-EA TILL NOP
3CAD-EA TILH NOP
*
* DEFINE BUFFERS
* -----
3CAE-EA ZBUF32 NOP
ZEND32 EQU ZBUF32+ZLEN
TBUF32 EQU ZEND32+1
TEND32 EQU TBUF32+TLEN
DBUF32 EQU TEND32+1
DEND32 EQU DBUF32+DLEN
ZBUF33 EQU DEND32+1
ZEND33 EQU ZBUF33+ZLEN
TBUF33 EQU ZEND33+1
TEND33 EQU TBUF33+TLEN
DBUF33 EQU TEND33+1
DEND33 EQU DBUF33+DLEN
ENDEND EQU DEND33+1
*
* --- END ASSEMBLY ---
*
TOTAL ERRORS = 0
PROGRAM LENGTH = 175 BYTES
*
* --- SYMBOL TABLE ---
*
ZSTART $00 ZEND $FF
TSTART $0300 TEND $03FF
DSTART $9600 DEND $BFFF
ZLEN $FF TLEN $FF
DLEN $29FF MOVER $3C38
ZBUF32 $3CAE ZEND32 $3DAD
TBUF32 $3DAE TEND32 $3EAD
DBUF32 $3EAE DEND32 $3FAD
ZBUF33 $68AE ZEND33 $69AD
TBUF33 $69AE TEND33 $6AAD
DBUF33 $6AAE DEND33 $6FAD
ALI $3C6F FROML $3C7E
FROMH $3C7F TILL $3CAC
TILH $3CAD TOL $3C81
TOH $3C82 RETAD $3C6D
MOVE $3C7D TFER $3C73
LLOAD $3C7A LHIAD $3C7B
INCCMP $3C89 ICNT01 $3C91
ICNT02 $3C99 IEND1 $3CA4
IEND2 $3CAB ENDEND $94AE

```

Beliebiges Rangieren zwischen zwei Disketten-Betriebssystemen für die Maschinen der Apple-Klasse erlaubt dieses Programm – darüber hinaus erschließt es ohne zusätzlichen Hardware-Aufwand die Vorzüge von DOS 3.3

gequälten „Syntax Error“, hatte aber, wie sich gleich darauf herausstellte, die angeordnete Umschaltanweisung trotz dieses Protestes dennoch einwandfrei ausgeführt.

Wer seine Maschine auf „offiziellen“ Wege auf DOS 3.3 umzurüsten gedenkt,

der muß bekanntlich zwei ROMs tauschen. Mit den neuen Bausteinen und der speziellen Boot-Diskette „BASICS – 13 SECTORS“ kann man zwar zurück ins alte Format, Vorbedingung alles dessen sind aber die neuen ROMs, an denen laut Hersteller- und Händler-Aussagen kein Weg vorbeiführt. Zu seiner Überras-

schung erlebte der Verfasser nun, daß auch mit alten ROMs ausgerüstete Controller-Karten nach anfänglichem Booten mit DOS 3.2 sehr wohl in der Lage waren, Disketten im neuen Format zu lesen, selbst mit 16 Sektoren pro Spur zu schreiben und sogar unformatierte Neu-Disketten mit einem lupenreinen DOS 3.3 zu initialisieren! Freilich zeigte sich, daß eine gesteigerte Empfindlichkeit gegenüber geringfügigen Geschwindigkeitsabweichungen der Laufwerke auftrat: Lesefehler kommen häufiger vor als mit der neuen Hardware. So wird man Double DOS Switch keineswegs als vollgültigen Ersatz für den Original-Umrüstsatz betrachten dürfen, sehr wohl aber sicherlich als ein taugliches Mittel, vor-

erst einmal ohne Hardware-Zusatzkosten erste Betriebserfahrungen mit der DOS-Version 3.3 zu sammeln.

Literatur

- [1] Joepgen, Hans-Georg: Ein (fast) ideales Speichermedium – das Disk-Operating-System des Euro-Apple. FUNKSCHAU 1980, Heft 6, Seiten 105...107.
- [2] Reynolds, William: Disassembling the DOS 3.2 MICRO 1979, Heft 17, Seiten 7...11. Micro Inc., Chelmsford, USA.
- [3] Luebbert, Prof. Dr. William F.: What's Where in the Apple. MICRO 1979, Heft 15, Seiten 30...36.
- [4] Joepgen, H.-G.: Mehr Komfort bei Apple-Disks. mc 1981, Heft 1, S. 64...66.

„home“ und „Cursor nach unten“ stehen, so geht das beim Apple einfach mit der VTAB-Anweisung, z. B. VTAB(3) für die dritte Bildschirmzeile.

Vorsicht ist bei Operationen wie AND und OR geboten. Beim Apple funktionieren sie nur nach IF so wie beim CBM, lassen aber keine arithmetischen Verknüpfungen zu. Während so beim CBM die Anweisung PRINT 15 AND 7 das Ergebnis 7 erbringt, errechnet der Apple-II nur 1. Denn AND und OR können bei ihm nur 1 für „wahr“ und 0 für „falsch“ ergeben. Währenddessen entspricht beim CBM auch die Wahrheitsaussage einem anderen Zahlenwert, nämlich -1, z. B. bei PRINT 2<3.

Der Apple-Befehl CALL entspricht dem CBM-Befehl SYS; allerdings ist bei CALL keine Parameterübergabe nach der Adresse möglich. Bei der Ausgabe von Zahlen oder bei ihrer Umwandlung in einen String mit STR\$ fügt der CBM am Anfang statt eines positiven Vorzeichens einen Leerraum und hinter der Zahl einen weiteren an, während der Apple-II auf das Hinzufügen von Leerräumen verzichtet.

Obwohl sich auf den ersten Blick die Basic-Interpreter von CBM und Apple-II sehr ähnlich sehen, gibt es beim Umschreiben von Programmen vom einen auf den anderen Rechner also eine Menge zu beachten. Fe.

Apple-Eigenheiten

Viele in mc abgedruckten Programme sind für Computer wie CBM, C-64, VC-20 oder CP/M-Basic-Rechner geschrieben. Für Apple-Anwender mit dem Applesoft-Basic-Interpreter ist es manchmal dann schwierig, solche Programme an den eigenen Rechner anzupassen, weil es doch eine ganze Reihe von Syntax-Unterschieden gibt.

Die Zeile

```
100 GET K$:IF K$="" THEN 100
```

würde beim Apple-II einfach lauten:

```
100 GET K$
```

Denn Applesoft hält beim GET-Befehl das Programm an und macht im Gegensatz zu anderen Computern übrigens auch den Cursor sichtbar. Will man das nicht, so kann man als Ersatz für den CBM-Basic-Befehl GET beim Apple folgende Routine verwenden:

```
100 A=PEEK(-16384):IF A>128 THEN  
K$="":RETURN
```

```
100 K$=CHR$(A-128):A=  
PEEK(-16368)
```

Ein weiteres Problem beim Apple-II ist das Fehlen besonderer Cursorbefehle, die man (wie beim CBM) in PRINT-Anweisungen aufnehmen könnte. Lediglich PRINT CHR\$(8) für Cursor nach links und PRINT CHR\$(10) für Cursor nach unten stehen zur Verfügung. Aber eine Lösung gibt es trotzdem: CALL-998 bewegt den Cursor eine Zeile nach oben,

CALL-1036 ein Zeichen nach rechts, und zwar ohne Löschen von Zeichen.

Der CBM-Befehl zum Löschen des Bildschirms (PRINT"Clr";) heißt beim Apple-II HOME. Statt PRINT"rvs" bzw. PRINT"rvs off" beim CBM muß man beim Apple INVERSE bzw. NORMAL schreiben, um Schriftzüge invers anzuzeigen.

Wenn in einem CBM-Programm nach PRINT nacheinander Anweisungen für

Listbare Autostart-Programme

Die im Bild abgedruckte 6502-Assembler-Routine befähigt den Apple-II und seine kompatiblen Abkömmlinge, Autostart-Programme von Kassette so einzulesen, daß der Autostart unterdrückt wird und auch ein Auflisten wieder möglich

ist. Statt dem gewöhnlichen LOAD-Befehl wird CALL 768 eingegeben. Nach dem zweiten Piepton befindet sich das Programm listbar im Speicher. Die Hilfsroutine ist beliebig verschiebbar.

Peter Engels

Mit diesem Hilfsprogramm kann man beim Apple-II Autostart-Programm wieder listen

0300-	20 F0 DB	JSR	\$DBF0
0303-	20 FD FE	JSR	\$FEFD
0306-	18	CLC	
0307-	A5 67	LDA	\$67
0309-	65 50	ADC	\$50
030B-	85 69	STA	\$69
030D-	A5 68	LDA	\$68
030F-	65 51	ADC	\$51
0311-	85 6A	STA	\$6A
0313-	A9 00	LDA	##00
0315-	85 D6	STA	\$D6
0317-	4C E0 D8	JMP	\$D8E0

Rudolf Hofer

MX-80 druckt Apple-Grafik

Grafikanwendungen bedeuten für den Programmierer meist doppelten Aufwand, wenn er das Ergebnis seiner Bemühungen nicht nur auf dem Bildschirm, sondern auch gedruckt bewundern will. Denn die Steuerbefehle für Plotter sind zwar den Bildschirmgrafik-Befehlen ähnlich, aber eben doch nicht mit ihnen identisch. Abhilfe schafft ein Programm, das einen beliebigen Bildinhalt zu Papier bringt. Im vorliegenden Fall gibt ein MX-80 (ohne Grafik-ROM) den Inhalt einer HGR-Seite des Apple aus.

Das Problem besteht darin, den Bildinhalt Punkt für Punkt abzutasten und auf das Papier mit genügender Auflösung zu

übertragen. Dies gestaltet sich zunächst schwieriger, als es den Anschein hat. Erstens ist der Bildspeicher des Apple

für hochauflösende Grafik sehr undurchsichtig organisiert. Und zweitens müssen jeweils sechs Punkte aus drei aufeinanderfolgenden Zeilen zu einem Grafikzeichen des MX-80 zusammengesetzt werden.

Die Adresse einer Bildschirmposition wird ermittelt

Bild 1 zeigt, wie der Bildschirm der beiden HGR-Seiten organisiert ist. Die 40 Spalten einer Zeile entsprechen 40 Adressen, die von links nach rechts in aufsteigender Reihenfolge geordnet sind. Jede einzelne Spalte wiederum besteht aus 7 Bildpunkten, den ersten 7 Bits des abgespeicherten Bytes. Die erste Adresse einer Zeile – im folgenden Basisadresse genannt – wird nach Bild 2 ermittelt. Sie ergibt sich, indem man die niederwertigen sechs Bits eines Zeilenzählers (oberste Zeile = null) an die in Bild 2 bezeichnete Position innerhalb der zwei Adreßbytes setzt. Je nach auszudruckender Bildschirmseite werden

```

**END OF PASS 1
**END OF PASS 2

0800      1  ;AUSGABE DER HGR-SEITE
0800      2  ;MIT DEM MX-80
0800      3  ;
0800      4  ZZ      EPZ 0          ; ZEILENZÄHLER
0800      5  SZ      EPZ 1          ; SPALTENZÄHLER
0800      6  BASIS   EPZ 2          ; BASISADRESSE
0800      7  XTEMP   EPZ 4
0800      8  ATEMP   EPZ 5
0800      9  MASKE   EPZ 6
0800     10  RCHAR   EPZ 7
0800     11  PRINT   EQU $C200      ; I ZCH AUSGEBEN
0800     12  ;WIRD DURCH DIE ANFANGSADRESSE
0800     13  ;DES EIGENEN DRUCKERTREIBERS
0800     14  ;ERSETZT
0800     15  ;
9000     16          ORG $9000
9000     17          OBJ $800
9000 A90F     18  PICT  LDA #$0F
9002 2000C2   19          JSR PRINT      ; SCHNALL
9005 A200     20          LDX #0
9007 8600     21          STX ZZ
9009 8601     22          STX SZ
900B     23  ;HAUPTPROGRAMMSCHLEIFE
900B 207690   24  MAIN  JSR PR7
900E E601     25          INC SZ
9010 A501     26          LDA SZ
9012 C924     27          CMP #$24
9014 90F5     28          BCC MAIN
9016 A900     29          LDA #0
9018 8501     30          STA SZ
901A A90D     31          LDA #$0D
901C 2000C2   32          JSR PRINT
901F E600     33          INC ZZ
    
```

Bild 3. Das Ausdruckprogramm: Die Anzahl der auszudruckenden Zeilen und Spalten kann geändert werden, indem man die entsprechenden Werte in die Speicherstellen \$9028 und \$9013 bringt. Um die HGR2-Seite auszudrucken, ist die Speicherstelle \$904B auf \$40 abzuändern. Das Unterprogramm PRINT muß ein Zeichen zum Drucker schicken; es beginnt normalerweise bei CN00, wobei N die Nummer des „Slots“ ist, in der das Drucker-Interface steckt

```

9021 E600     34          INC ZZ
9023 E600     35          INC ZZ
9025 A500     36          LDA ZZ
9027 C9A0     37          CMP #160
9029 90E0     38          BCC MAIN
902B 60       39          RTS
902C         40  ;
902C         41  ;BASISADR. AUS ZZ UND
902C         42  ;SZ BERECHNEN
902C         43  ;BYTE HOLEN UND 2 BIT
902C         44  ;RECHTSBÄNDIG IN AKKU
902C         45  ;(X)=ZAHL DER SCHIEBEOP.
902C A500     46  CALC  LDA ZZ
902E         47  ;LOW BYTE
902E 2908     48          AND #%00001000
9030 0A       49          ASL
9031 0A       50          ASL
9032 0A       51          ASL
9033 0A       52          ASL
9034 8502     53          STA BASIS
9036         54  ;HIGH BYTE
9036 A500     55          LDA ZZ
9038 293F     56          AND #%00111111
903A 8505     57          STA ATEMP
903C 4605     58          LSR ATEMP
903E 4605     59          LSR ATEMP
9040 4605     60          LSR ATEMP
9042 4605     61          LSR ATEMP
9044 0A       62          ASL
9045 0A       63          ASL
9046 291C     64          AND #%00011100
9048 0505     65          ORA ATEMP
904A         66  ;BEGINN BEI $2000
904A 0920     67          ORA #%00100000
904C 8503     68          STA BASIS+1
904E         69  ;SPALTENADRESSE
904E         70  ;JE NACH BLOCK
904E         71  ;NACH Y
904E 8604     72          STX XTEMP
9050 A501     73          LDA SZ
9052 A600     74          LDX ZZ
9054 E0B0     75          CPX #$80
9056 9005     76          BCC EINS
9058 18       77          CLC
    
```


Das Apple-Sonderheft

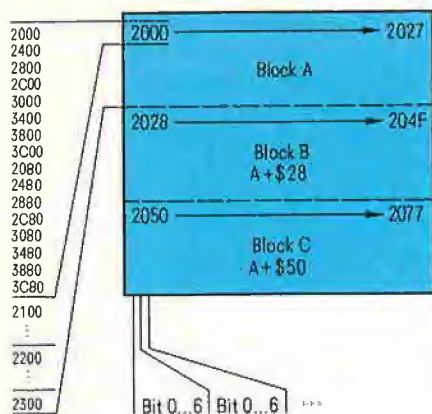


Bild 1. Aufteilung der HGR-Seite beim Apple: Hier ist die erste Seite dargestellt, die bei der 160. Zeile endet. Die zweite HGR-Seite hat 192 Zeilen. Sie beginnt bei Adresse \$4000 und ist ansonsten gleich organisiert

Bildschirmadresse	Bit	12	11	10	9	8	7	6	5	4	3	2	1	0
6-Bit- Zeilenzähler*	Bit	2	1	0	5	4	3	X	X	X	X	X	X	X
7-Bit-Spaltenzähler	Bit	X	X	X	X	X	X	6	5	4	3	2	1	0

*Bit 6 und 7 nicht relevant
Alle Angaben hexadezimal

Block A: → 00....27
Block B: +28 → 28....4F
Block C: +50 → 50....77

Bild 2. Ermittlung der relativen Adresse aus Zeilenposition (ab null) und Spaltennummer (ab null) durch Umstellen von Bits. Die Spaltennummer bezieht sich auf ein Byte, dessen erste sieben Bits jeweils sieben Bildpunkte (Bit 0 ist links) repräsentieren. Zur relativen Adresse wird \$2000 (HGR-Seite 1) bzw. \$4000 (HGR-Seite 2) addiert

```

9059 6950    78      ADC #$50
905B D007    79      BNE RET
905D E040    80      CPX #$40
905F 9003    81      BCC RET
9061 18      82      CLC
9062 6928    83      ADC #$28
9064 A8      84      RET
9065 B102    85      LDA (BASIS),Y
9067 2506    86      AND MASKE
9069 A604    87      LDX XTEMP
906B E000    88      NORM CPX #0
906D F004    89      BEQ BACK
906F 4A      90      LSR
9070 CA      91      DEX
9071 D0F8    92      BNE NORM
9073 A604    93      BACK LDX XTEMP
9075 60      94      RTS
9076         95      ; -----
9076         96      ; 7 ZEICHEN AUSGEBEN
9076 A200    97      PR7 LDX #0
907B 8607    98      STX RCHAR
907A A903    99      LDA #00000011
907C 8506   100      STA MASKE
907E 20C890 101      JSR ASS ; 1. ZCH
9081 A202   102      LDX #2 ; 2. ZCH
9083 20C890 103      JSR ASS ; 3. ZCH
9086 A204   104      LDX #4 ; 4. ZCH
9088 20C890 105      JSR ASS ; 5. ZCH
908B A200   106      LDX #0 ; 6. ZCH
908D 20F390 107      JSR CONN ; 7. ZCH
9090 E600   108      INC ZZ
9092 20F390 109      JSR CONN
9095 E600   110      INC ZZ
9097 20F390 111      JSR CONN
909A C600   112      DEC ZZ
909C C600   113      DEC ZZ
909E E601   114      INC SZ
90A0 A903   115      LDA #00000011
90A2 8506   116      STA MASKE
90A4 20C890 117      JSR ASS ; 4. ZCH
90A7 A202   118      LDX #2 ; 5. ZCH
90A9 20C890 119      JSR ASS ; 6. ZCH
90AC A204   120      LDX #4 ; 7. ZCH
90AE 20C890 121      JSR ASS ; 8. ZCH
90B1 A206   122      LDX #6 ; 9. ZCH
90B3 20C890 123      JSR ASS ; 10. ZCH
90B6 20FF90 124      JSR KORR
90B9 E600   125      INC ZZ
90BB 20FF90 126      JSR KORR
90BE E600   127      INC ZZ
90C0 20FF90 128      JSR KORR
90C3 C600   129      DEC ZZ
    
```

```

90C5 C600    130      DEC ZZ
90C7 60      131      RTS
90C8         132      ; -----
90C8         133      ; 1 ZEICHEN ZUSAMMENBAUEN
90C8         134      ; UND AUSGEBEN
90C8 202C90 135      ASS JSR CALC
90CB 200C91 136      JSR SHIF
90CE E600    137      INC ZZ
90D0 202C90 138      JSR CALC
90D3 200A91 139      JSR SHIF2
90D6 E600    140      INC ZZ
90D8 202C90 141      JSR CALC
90DB 200B91 142      JSR SHIF4 ; REL. ZCH
90DE A9A0    143      LDA #$A0
90E0 18      144      CLC
90E1 6507    145      ADC RCHAR ; ABS. ZCH
90E3 2000C2 146      JSR PRINT
90E6 A900    147      LDA #0
90EB 8507    148      STA RCHAR
90EA C600    149      DEC ZZ
90EC C600    150      DEC ZZ
90EE 0606    151      ASL MASKE
90F0 0606    152      ASL MASKE
90F2 60      153      RTS
90F3         154      ; -----
90F3         155      ; BIT 6 IN NACHSTES BYTE
90F3 202C90 156      CONN JSR CALC
90F6 2A      157      ROL
90F7 2A      158      ROL
90FB C8      159      INY
90F9 B102    160      LDA (BASIS),Y
90FB 2A      161      ROL
90FC 9102    162      STA (BASIS),Y
90FE 60      163      RTS
90FF         164      ; -----
90FF         165      ; BIT 6 WIEDER KORRIGIEREN
90FF 202C90 166      KORR JSR CALC
9102 B102    167      LDA (BASIS),Y
9104 4A      168      LSR
9105 9102    169      STA (BASIS),Y
9107 60      170      RTS
9108         171      ; -----
9108 0A      172      SHIF4 ASL
9109 0A      173      ASL
910A 0A      174      SHIF2 ASL
910B 0A      175      ASL
910C 05Q7    176      SHIF ORA RCHAR
910E 8507    177      STA RCHAR
9110 60      178      RTS
9111         179      ; -----
9111         180      PAU
    
```



Bild 4. Ausdruck einer Bezierkurve [1] mit den Stützpunkten 0,0; 140,0; 0,160 und 140,80. Die Breite entspricht also dem halben Bildschirm (Verkleinerungsfaktor 50 %)

die Bits 15...13 auf 010 (\triangleq Startadresse \$4000) oder 001 (\triangleq Startadresse \$2000) gesetzt. Die Bits 6...0 legen einerseits die Spalte fest, andererseits einen von drei Bildschirmbereichen. Repräsentieren sie einen Wert zwischen 0 und \$27, so liegt eine Bildschirmadresse im Block A (siehe Bild 1) vor. Für Block B liegt der Wert zwischen \$28 und \$4F, und für Block C ist er größer als \$4F. Diese zusätzliche Information ist notwendig, weil die sechs maßgeblichen Bits des Zeilenzählers nur 64 Zeilen festlegen können. Das in Bild 3 abgedruckte Programm kann mit CALL 36864 von BASIC aus aufgerufen werden (48-K-System). Zuvor ist mit HIMEM: 36863 der BASIC-Speicherbereich zu begrenzen. Will man die zweite HGR-Seite ausdrucken, dann ist das Byte \$20 in der Speicherstelle \$904B auf \$40 zu ändern (z. B. mit POKE 37899, 64). Bild 4 zeigt einen Beispielausdruck. Zu beachten ist, daß der MX-80 nur maximal 264 Bildpunkte in horizontaler Richtung zu Papier bringen kann. Die Grafikseite ist jedoch 280 Punkte breit.

Literatur

[1] Andree, Hans-Joachim: Kurvenentwicklung auf Bildschirm und Plotter. mc 1982, Heft 5, S. 42...44.

Apple-Kniffe

Wer seinen Apple-II nach mc 6/1983, Seite 74, auf Kleinschrift umrüstet, erlebt beim ersten Einschalten eine Überraschung: Sein Computer meldet sich mit „Apple ÄÜ“. Der Änderungsvorschlag aus mc 12/1983 läßt sich beim Original-Apple je nach Version nicht so leicht durchführen, da die Apple-ROMs u. U. andere Chip-Select-Signale besitzen. (Besitzer von Apple-Nachbauten sind da im Vorteil, allerdings ist deren Zeichengenerator je nach Typ u. U. auch anders organisiert.) Beim Original-Apple kann eine Änderung des Zeichengenerators Abhilfe schaffen, wenn man keine

```
00D8 : 3E 30 30 30 30 30 3E 00
00E0 : 00 20 10 08 04 02 00 00
00E8 : 3E 06 06 06 06 06 3E 00
02D8 : BE B0 B0 B0 B0 B0 BE 80
02E0 : 80 A0 90 88 84 82 80 80
02E8 : BE 86 86 86 86 86 BE 80
06D8 : 3E 30 30 30 30 30 3E 00
06E0 : 00 20 10 08 04 02 00 00
06E8 : 3E 06 06 06 06 06 3E 00
```

Bild 1. Änderung des deutschen Zeichengenerators beim Original-Apple für ASCII-Zeichen statt deutscher Umlaute

deutschen Umlaute benötigt: Bild 1 zeigt den entsprechenden EPROM-Ausschnitt. Umlaute können ja über die Apple-Tastatur ohnehin nicht eingegeben werden.

Obwohl das Apple-Monitorprogramm

Klein- in Großbuchstaben umwandelt, sollten Programmeingaben stets in Stellung Großschreibung des in mc 6/1983 vorgeschlagenen Umschalters erfolgen, sonst sind die Escape-Funktionen zum Editieren nur mit gedrückter Shift-Taste erreichbar.

Ohne Betriebssystem-Änderung (vgl. mc 12/1983, Seite 53) sind Kleinbuchstaben in ein Basic-Programm nur mit erheblichen Umwegen zu schmuggeln: Dazu muß per Programm Zeichen für Zeichen umgewandelt oder mit CHR\$() definiert werden. Durch ein Basic-Programm (Bild 2) kann aber ein entsprechender String aufgebaut werden, da GET Kleinbuchstaben annimmt. Ein Backspace wird hierbei als Delete ausgeführt; TAB ist unwirksam. Der Name eines Disketten-Files sollte tunlichst so nicht erzeugt werden, da er mit Kleinbuchstaben in den Disketten-Katalog mit aufgenommen würde und durch direkt eingegebene Disk-Kommandos nicht mehr aufgerufen werden kann.

Die Änderung nach mc 3/1983, Seite 49, ist u. U. für einige Überraschungen gut.

Da der Apple-II bei vielen Gelegenheiten den AN2-Ausgang des Game Connectors auf High-Pegel setzt, kommt man leicht unverhofft zu Blockgrafik auf dem MX-80, wenn man Schrift erwartet. Unter anderem passiert dies auch bei einer Garbage Collection. Eine Alternative ist es, Bit 7 des Drucker-Interface über einen Inverter anzusteuern und die Befehle zu tauschen (POKE -16291,0 für Buchstaben und POKE -16292,0 für Grafik).

Dr. Rolf Blasberg

```
1000 REM T$ MIT KLEINBUCHSTABEN
1002 T$ = "" : L = 0
1004 GET B$
1006 IF B$ = CHR$ (13) THEN RETURN
1008 IF ASC (B$) = 8 THEN 1014
1010 IF B$ < " " THEN 1004
1012 PRINT B$ ; : T$ = T$ + B$ : L = L + 1 : GOTO 1004
1014 IF L < 1 THEN 1004
1016 PRINT B$ ; " " ; B$ ; : IF L = 1 THEN T$ = "": GOTO 1020
1018 T$ = LEFT$ (T$, L-1)
1020 L = L - 1 : GOTO 1004
```

Bild 2. Basic-Programm zum Eingeben von Strings mit Kleinbuchstaben

Hans-Georg Joepgen

Präzises Paginieren per „PUT“

Es gibt drei gute Gründe für die Vorstellung des im folgenden besprochenen Apple-Programmpaketes „Printer Utilities“ (PUT): Zum einen erhalten mit Hilfe von PUT selbst einfachere Drucker Fähigkeiten, wie man sie sonst vornehmlich bei Spitzengeräten findet. Zum zweiten wird vorgeführt, daß die bisher kaum angewandte Unterstützung maschinensprachlicher Programmsegmente von einer Hochsprache aus interessante Vorteile bietet – und schließlich zeigt der Beitrag noch auf, daß der Datenfluß zwischen dem Apple-Betriebssystem und DOS 3.2 bequem über Eigenbau-Betriebssystemerweiterungen geleitet werden kann, was vielerlei Möglichkeiten eröffnet.

Was bei leistungsfähigeren Anlagen der Mittleren Datentechnik und erst recht in Großrechnersystemen seit langem zu den Selbstverständlichkeiten zählt, intelligentes Drucker-Verhalten beim Paginieren (Seiteneinteilung) nämlich, darauf müssen Mikrocomputer-Anwender mit ihren zumeist schlichteren Druckern in aller Regel verzichten. Will man beim Ausdrucken mehrseitiger Tabellenwerke Zwischenüberschriften und Seitenzahl-Angaben als Seitenkopf einschließlich freien Raumes ober- und unterhalb

```
P U T  P R I N T E R  U T I L I T I E S
=====

GETTING:
-----

      MASTER FILE (1)

      USERS  FILE (2)

      NO NEW FILE (3) ?

P U T  STATUS:  OK.-

-----

(1) LINES PER PAGE: ..... 72
(2) HEAD LINE LOCATION: ..... 8
(3) START OF TEXT: ..... 10
(4) END OF TEXT: ..... 60
(5) HEAD LINE TEXT: .....

      "++ PUT OVERLAY V1.4 (JOE.) "
      -----
      +++  CHANGE  (#)
      +++  SAVE    (S)
      +++  RESTART (R)
      +++  QUIT    (Q) ? Q

      BYE.-
```

Bild 1. „PUT OVERLAY“ ist menüorientiert: Das bedeutet bequeme Kommandoabgabe und übersichtliches Melden der aktuellen Parameter

Bild 2. Das Basic-Programm „PUT OVERLAY“, hier dargestellt mit Hilfe von „PUT“ selbst. Es erlaubt die Überprüfung und Modifikation des Hauptprogrammes PUT

```
++ PUT OVERLAY V1.4 (JOE.) * PAGE 1
```

```
1  REM -----
2  REM  P U T  OVERLAY
3  REM -----
4  REM
5  REM  VERSION 1.4
6  REM
7  REM  (PART OF PRINTER UTILITIES PACKAGE)
8  REM
9  REM  (HANS-GEORG JOEPGEN, 251080,311280,170181)
10 :
11 SUM = 32308:MSTART = 36666:MFIN = 36928:TBASE = 36983:CBASE = 36979: POKE
    33,40
12 HOME : VTAB 5: HTAB 10: PRINT " P U T  P R I N T E R  U T I L I T I E S"
13 HTAB 10: PRINT " =====": PRINT : PRINT
14 PRINT " GETTING:"
15 PRINT " -----"
16 PRINT
17 PRINT "      MASTER FILE (1)"
18 PRINT
19 PRINT "      USERS  FILE (2)"
20 PRINT
21 PRINT "      NO NEW FILE (3) ? "
22 GET A$:A = VAL (A$): IF (A < 1) OR (A > 3) THEN PRINT CHR$ (7): GOTO
    22
23 IF A = 3 THEN 27
24 IF A = 2 THEN A$ = "USERS FILE": GOTO 26
25 A$ = "MASTER FILE"
26 PRINT : PRINT CHR$ (4)"BLOAD PUT,"A$
27 GOSUB 38: REM S & C-TABLE
28 GET A$: PRINT A$:
29 IF FLAG THEN 31
30 IF A$ = "S" THEN PRINT CHR$ (4)"BSAVEPUT.USERS FILE,A$B34,L$168": GOTO
    27
31 IF A$ = "Q" THEN HOME : PRINT " ": PRINT " ": PRINT " BYE.-": END
32 IF A$ = "R" THEN 12
33 N = VAL (A$): IF (N < 1) OR (N > 5) OR (N < > INT (N)) THEN PRINT CHR$
    (7): GOTO 28
34 ON N GOSUB 73,73,73,73,79: GOTO 27
35 REM
36 REM  STATUS AND COMMAND TABLE
37 REM -----
38 PRINT " ": PRINT " ": PRINT " ": HOME : PRINT " ": PRINT " P U T  STAT
    US: "
39 FLAG = 0: FOR I = 2 TO 4: IF ( PEEK (CBASE + I)) = > ( PEEK (CBASE + I
    + 1)) THEN FLAG = 1: NEXT
40 IF ( PEEK (CBASE + 4)) > ( PEEK (CBASE + 1)) THEN FLAG = 1
41 CHECK = 0: FOR COUNTER = MSTART TO MFIN:CHECK = CHECK + PEEK (COUNTER)
    : NEXT COUNTER: IF CHECK < > SUM THEN FLAG = 1
42 IF FLAG THEN FLASH : PRINT CHR$ (7)"E R R O R": NORMAL : GOTO 44
```

des Text-Blockes sehen, so muß dieses im Benutzerprogramm jeweils gesondert vorgesehen sein. Drucker, die – wie der HP 9871 zum Beispiel – die genannten Anforderungen wenigstens teilweise erfüllen, kosten mehr als das ganze übrige voll ausgebaute Computersystem einschließlich mehrerer Diskettenstationen und bieten dennoch nicht immer die Möglichkeit eines selbsttätigen Kopfausdrucks jeweils zu Seitenbeginn.

Als wünschenswert erschien deswegen eine Betriebssystem-Erweiterung für die Maschinen der Apple-Klasse, die – einmal initialisiert – für den Benutzer fortan „unsichtbar“ bleibt, wenig Speicher- raum belegt und zur Zusammenarbeit mit beliebigen Benutzerprogrammen fähig sein sollte, ohne daß in diesen Benutzerprogrammen besondere Vorkehrungen zur Kooperation mit der Erweiterung erwartet werden durften.

Das Software-Paket „PUT“, entwickelt und erprobt auf einer mit 48 KByte RAM und den jüngsten DOS-Versionen ausgerüsteten Maschine ITT 2020, erfüllt die Anforderungen und korrigiert darüber

hinaus noch gewisse Unvollkommenheiten, die sich aus dem Zusammenwirken der Apple-Centronics-Interface-Karte mit Druckern der Typenreihe 779 und den nach gleichem Standard arbeitenden Konkurrenzprodukten ergeben.

Drucker-Intelligenz in nur 360 Byte

Der Widerspruch zwischen den beiden einander eigentlich wechselseitig ausschließenden Anforderungen an PUT, hoher Bedienungskomfort bei der Parameter-Wahl einschließlich ausgedehnter Plausibilitätskontrollen auf der einen Seite, und geringe Speicherplatzbelegung auf der anderen, wurde wie folgt gelöst: Das Programmsegment, das die eigentliche Arbeit macht (PUT.USERS FILE) und dessen Umfang einschließlich aller benötigten Buffer auf 360 Byte beschränkt werden konnte, residiert unterhalb des Disketten-Betriebssystems im RAM-Bereich und sorgt durch Pointer-Manipulation dafür, daß es vom Rechner vor Überschreiben geschützt bleibt.

Jede Rechner-Ausgabe führt nun über PUT.USERS FILE, die hierbei selbsttätig erkennt, ob sie den Drucker zum Fortschalten auf den nächsten Seitenanfang bewegen, Leerzeilen einfügen oder einen Seitenkopf-Zwischentitel ausdrucken soll. Da das Centronics-Interface von Apple-II und ITT-2020 beim Kommando „CR“ das „Line Feed“ vergißt, sofern der Drucker-Buffer ansonsten leer ist, behebt PUT.USERS FILE diesen Mangel gleich mit. Darüber hinaus dekodiert und befolgt PUT.USERS FILE auch das ASCII-Kommando „FORM FEED“, das Vorrück- ken auf die nächste Textseite in solchen Fällen, wenn der vorgegebene Textblock noch nicht gefüllt ist.

All dies in 360 Byte – da bleibt kein Raum zu einer komfortablen, menü- orientierten Parameter-Eingabe und Operator-Kontrolle. Es wurde deswegen eine Arbeitsteilung vorgenommen. Ausgehend von der Tatsache, daß beim „Einrichten“ von PUT, der Anpassung an Benutzerwünsche einschließlich der Texteingabe für die Seitenkopf-Überschriften, ein Benutzer-Programm in aller Regel noch nicht geladen ist, wurden diese Aufgaben einem Basic-Programm „PUT OVERLAY“ übertragen. Auf der PUT-Diskette liegt eine Aufzeichnung „PUT.MASTER FILE“ vor – eine betriebsbereite PUT-Version mit folgenden festen Parametern: Zeilenzahl pro Seite: 72, Überschrift in Zeile 2, Textbeginn in Zeile 5, Textende in Zeile 68 und Wort- lauf der Seitenkopf-Überschrift: Eine Gruppe von Leerzeichen, denen das Wort „PAGE“ plus die Seitennummer folgt.

Basic-Programm kontrolliert Maschinencode

Will man es bei dieser Grundeinstellung bewenden lassen, so reicht es aus, PUT.MASTER FILE zu laden und dann zu initialisieren. Dies geschieht von Basic aus mit dem Befehl CALL 36660, vom Monitor aus mit der Go-Anweisung 8F34G. Hat man, wie weiter unten beschrieben, bereits Modifikationen eingegeben und wünscht sich des dergestalt veränderten PUT-Segments zu bedienen, so lädt man PUT.USERS FILE und verfährt dann weiter wie oben.

Die Produktion einer neuen USERS FILE erfolgt durch „PUT OVERLAY“. Dieses menü-orientierte und somit in hohem Maße benutzerfreundliche Kontrollprogramm erlaubt wahlweise, von der MASTER FILE, der USERS FILE oder einer bereits im Speicher befindlichen PUT-

```

++ PUT OVERLAY V1.4 (JOE.) * PAGE 2

43 PRINT " OK.-"
44 PRINT : PRINT " -----":N = 1
45 PRINT : PRINT
46 PRINT " ": PRINT " (1) LINES PER PAGE: ..... ";
47 GOSUB 67
48 PRINT " ": PRINT " (2) HEAD LINE LOCATION: ..... ";
49 GOSUB 67
50 PRINT " ": PRINT " (3) START OF TEXT: ..... ";
51 GOSUB 67
52 PRINT " ": PRINT " (4) END OF TEXT: ..... ";
53 GOSUB 67
54 PRINT " ": PRINT " (5) HEAD LINE TEXT: ..... ": PRINT " "
55 GOSUB 85
56 PRINT " -----"
57 PRINT :
58 PRINT "      +++   CHANGE   (#)"
59 IF FLAG THEN 61
60 PRINT "      +++   SAVE   (S)"
61 PRINT "      +++   RESTART (R)"
62 PRINT "      +++   QUIT   (Q) ? ";
63 RETURN
64 REM
65 REM GET DATA FROM FILE
66 REM -----
67 A$ = STR$ ( PEEK (CBASE + N))
68 IF LEN (A$) < 3 THEN A$ = " " + A$: GOTO 68
69 PRINT A$:N = N + 1: RETURN
70 REM
71 REM MODIFY C BYTES
72 REM -----
73 HTAB 32: VTAB (2 * N + 6): PRINT "      ": HTAB 32: INPUT " ":A
74 IF (A < 1) OR (A > 255) OR (A < > INT (A)) THEN PRINT ".CHR$ (7): GOTO
73
75 POKE (CBASE + N),A: RETURN
76 REM
77 REM MODIFY T BYTES
78 REM -----
79 FOR N = 1 TO 28: POKE (TBASE + N),160: NEXT : GOSUB 85
80 VTAB 18: HTAB 6: INPUT A$: IF LEN (A$) > 27 THEN PRINT CHR$ (7): GOTO
79
81 FOR N = 1 TO LEN (A$): POKE (TBASE + N), ASC ( MID$ (A$,N,1)) + 128: NEXT
: RETURN
82 REM
83 REM SHOW HEAD LINE CONTENTS
84 REM -----
85 VTAB 18: HTAB 6: PRINT CHR$ (34): FOR N = 1 TO 27: PRINT CHR$ ( PEEK -
(TBASE + N) - 128): NEXT : PRINT CHR$ (34): RETURN

```


Das Apple-Sonderheft

Version auszugehen, diese zu ändern und, auf Wunsch, als neue USERS FILE auf Diskette abzuliegen. Plausibilitätskontrollen, Prüfsummentests und Vorkehrungen zum Schutze der durch Schreibvorgänge unerreichbaren MASTER FILE machen diese Lösung weitgehend narrensicher.

Das Programm OVERLAY unter der Lupe

Bild 1 zeigt, wie sich PUT OVERLAY meldet. Im vorgeführten Beispiel wurde Taste 2 betätigt, also eine USERS FILE abgerufen. Die Mitteilung „PUT STATUS: OK.“ zeigt an, daß in der File kein Prüfsummen-Fehler vorliegt und die gewünschten Parameter nicht im Widerspruch zu einander stehen. Es folgt die Aussage der Parameter „Anzahl der Zeilen pro Seite insgesamt“, „Zeilennum-

mer der Seitenkopf-Überschrift“, „Textbeginn“ und „Textende“. Anschließend meldet PUT OVERLAY den Inhalt des Zwischenüberschrift-Buffers, hier der für Bild 2 verwendete Text. Eine zweite Menü-Tafel bietet sodann folgende Möglichkeiten an: Änderung von Parametern oder Text, Speichern des Ist-Standes als neue USERS FILE auf Diskette, einen Neustart zur Rückkehr in die erste Menü-Tafel oder Verabschiedung von PUT OVERLAY. Im letzteren Fall bleiben die gegebenenfalls vorgenommenen Änderungen im Speicher und stehen nach Initialisierung per Call- oder Go-Befehl betriebsbereit zur Verfügung. Sie gehen nach Neuinitialisierung von DOS jedoch verloren.

Nun ein Blick in den jeweiligen Quellcode von PUT OVERLAY (Bild 2) und PUT MASTER FILE (Bild 3), ersterer bereits mit PUT gelistet. Da durch die häufige Wahl selbsterklärender Varia-

blen-Namen und gelegentliche Kommentare in den Listings für Erläuterung Sorge getragen ist, kann die Besprechung kurz gehalten werden: Zeile 11 des Basic-Programms lädt SUM mit der Prüfsumme über alle Bytes des MASTER FILE-Bereichs von MSTART (Masterfile-Start) bis MFIN (Masterfile-Finis), die auch für eine funktionsbereite USERS FILE zutrifft. TBASE und CBASE (TEXT BUFFER BASE und COUNTERS BUFFER BASE) bezeichnen Pufferbereichs-Anfänge. Ausgabe von Menü-Tafeln und Einholen von Benutzerkommandos erfolgen im Zeilenbereich bis 34, dem Hauptprogramm. In den Unterprogrammen bedarf die Variable FLAG der Erläuterung, die hier als Speicher für eine Boolesche Größe benutzt wird und nur dann gesetzt ist, wenn PUT OVERLAY auf Fehler stößt. Darüber hinaus: Keine Besonderheiten, die einer Programm-Analyse im Wege stünden.

```
* PUT MASTER FILE 170181.TXT
*
*****
* - P U T -
* *****
* PRINTER UTILITIES
* H.- G. JOEPGEN,
* VERSION 1.4
* *****
*
* DEFINITIONS
* -----
SLOT EQU 1
START EQU $B34 ; 3660 DEZ
DOSEX EQU $A53 ; DOS EXIT
COUT EQU $FDE
PRENT EQU 256*SLOT+$C002
HIMML EQU 115
HIMMH EQU 116
FF EQU $8C ; CNTRL L
CR EQU $8D
SP EQU $A0
*
* INIT
* -----
ORG START
LDA #START/256
STA HIMMH
LDA #START
STA HIMML
JSR CPRINT
ASC " * PUT "
ASC "CONNECTED.-"
DFB CR,$EA
LDA #PUT
STA DOSEX
LDA #PUT/256
STA DOSEX+1
LDA #0
STA CHFLAG
LDA #1
STA LINC
STA PAGC
RTS
*
* PUT ROUTINES
* -----
PUT STX XSAVE
STY YSAVE
STA ASAVE
JSR EMPTY
CMP #FF
BEQ FFEED
CMP #CR
BEQ CRPUT
LDY #1
STY CHFLAG
PUTEND LDA ASAVE
LDX XSAVE
LDY YSAVE
JMP PRENT
* DO CARRTN AND EXIT
CRPUT JSR UPDAT
JSR INSERT
LDY #0
STY CHFLAG
JMP PUTEND
*
* FORM FEED
* -----
FFEED LDA LINC
CMP TSTART
BEQ FFEND
JSR DOCR
JMP FFEED
FFEND JMP PUTEND
*
* RANGE CHECK
* -----
RCHECK LDA LINC
CMP TSTART
BCC OUTSID
CMP TEND
BCC OUTSID
RTS :INSIDE, C CLEAR.
OUTSID SEC
RTS :OUTSIDE, C SET.
*
* UPDATE COUNTERS
* -----
UPDAT INC LINC
LDA LINC
CMP LINES
BNE UPDAT1
LDA #1
STA LINC
LDA PAGC
SED
CLC
ADC #1
STA PAGC
CLD
UPDAT1 LDA LINC
CMP HLPOS
BNE UPDEND
JSR PHEAD
UPDEND RTS
*
* INSERT PAGE HEADLINE
* -----
PHEAD LDX #0
PHEAD1 CPX #36
BEQ PHEAD2
LDA PTEXT,X
JSR PRENT
INX
JMP PHEAD1
PHEAD2 LDA PAGC
LSR A
LSR A
LSR A
LSR A
BEQ PHEAD3
ORA #$B0
JSR PRENT
PHEAD3 LDA PAGC
AND #$0F
ORA #$B0
JSR PRENT
JSR DOCR
RTS
*
* INSERT DUMMY LINES
* -----
EMPTY JSR RCHECK
BCC EMPEND
JSR DOCR
JMP EMPTY
EMPEND LDA ASAVE
RTS
*
* INSERT SP IF CHFLAG OFF
* -----
INSERT LDY CHFLAG
BNE INSEND
LDA #SP
JSR PRENT
LDY #1
STY CHFLAG
INSEND RTS
*
* DO CARRTN AS SUBROUT
* -----
DOCR JSR UPDAT
JSR INSERT
LDA #CR
JSR PRENT
LDY #0
STY CHFLAG
RTS
*
* INCLUDED MACRO: PRINT *
* -----
* VERSION: JULY 20TH, 1980
*
HPRINT JSR $FC58 HOME
CPRINT JSR $FD5E CROUT
PRINT PLA
STA PCODE+1
PLA
STA PCODE+2
PRLOOP JSR PSUBRT
PCODE DFB $AD,0,0
CMP #$EA DONE?
BEQ PNTEND YES
JSR $FDED NO,DO
JMP PRLOOP REPEAT
PNTEND JSR PSUBRT UPDATE
JMP (PCODE+1) =RTS
PSUBRT INC PCODE+1 IND ADR
BNE PSBEND
INC PCODE+2 IS MSBYT
PSBEND RTS PRINT SUBR END.-
*****
*
* FLAGS, COUNTERS, DATA
* -----
LINC NOP LINES COUNTER
PAGC NOP PAGES COUNTER
CHFLAG NOP IF RESET, INSERT
ASAVE NOP SAVE AREA
XSAVE NOP SAVE AREA
YSAVE NOP SAVE AREA
LINES DFB 72
HLPOS DFB 2
TSTART DFB 5
TEND DFB 68
*
* PAGE HEADLINE TABLE
* -----
PTEXT ASC " "
ASC " "
ASC " "
ASC " "
ASC " * PAGE "
*
*****
<<END-OF-FILE>>
```

Bild 3. Listing des Quelltextes, aus dem die Objekt-Code-Aufzeichnung „PUT MASTER FILE“ entsteht, die unmittelbar eingesetzt oder als Kopiervorlage für eine an Benutzer-Änderungswünsche angepaßte „PUT USERS FILE“ benutzt wird

Eingeschmuggelt zwischen DOS und Monitor

Wenn die Maschinen der Apple-Klasse unter einem Disketten-Betriebssystem von der Art des DOS 3 laufen und ein Steckplatz als Datensinke initialisiert wird, dann wird durch entsprechendes Setzen von Pointern im Zero-Page-Bereich des Monitors und im DOS selbst der Datenstrom auf seinem Weg vom Entstehungsort nach DOS und von dort zurück in den Monitor aufgetrennt und eine „Umleitung“ über den Steckplatz geschaltet.

Genau dort nun greift PUT ein und trennt ein weiteres Mal auf, setzt sich gewissermaßen vor die Steckplatz-Karte und kontrolliert, was „durchkommt“ – um erforderlichenfalls die sendende Datenquelle zu stoppen und seinerseits Line-Feed-Zeichen, den Carriage-Return-Code oder ganze Überschrift-Zeilen einzufügen. Zum Verständnis des in Bild 3 gezeigten Listings sind Erläuterungen zu den Blöcken DEFINITIONS und FLAGS, COUNTERS, DATA vonnöten: SLOT ist die Nummer des Steckplatzes mit der Interface-Karte zum Drucker, DOSEX der DOS-Ausgangs-Vektor, PRENT (PRINT ENTRY) die Übergabe-Adresse zur Interface-Karte in Steckplatz SLOT. Der Doppelbyte-Pointer HIMEML, HIMEMH schafft eine „magische Grenze“ für Palsoft und hindert diesen Basic-Interpreter der Maschine 2020 an Beschädigungen von PUT. Die Labels FF, CR und SP bezeichnen die Codes für Formfeed, Carriage Return und Space in Apple-Notation: ASCII-Wert plus 128. LINC (Lines Counter) dient als Zeilenzähler, PAGC (Pages Counter) enthält die jeweils aktuelle Seitennummer. Der Inhalt von CHFLAG (Character Flag) verhilft PUT zur Fähigkeit, erforderlichenfalls ein Linefeed einem Carriage-Return-Befehl voranzustellen.

Um, wie gefordert, PUT für den Benutzer „unsichtbar“ zu machen, das heißt, außerhalb des eigentlichen PUT-Pflichtenkatalogs identisches Verhalten der Maschine in „PUT-Betrieb“ und „PUT-loser Betriebsart“ sicherzustellen und Randeffekte auszuschalten, stellt PUT jeweils am Ende einer seiner Eingreifaktionen den Original-Register-Zustand der CPU wieder her – dazu bedient es sich der Speicherstellen ASAVE, XSAVE und YSAVE als Zwischenlager. Es schließt sich der Buffer-Bereich für Variable und den Seiten-Überschrift-Text an: Mit Hilfe dieser Anmerkungen und der Listing-Kommentare läßt sich der Ablauf der Dinge schön studieren.

Betriebserfahrungen und Erweiterungsmöglichkeiten

Die vorgestellte PUT-Version hat sich in vielerlei Betriebssituationen als untadeliger Partner von DOS 3.2 über 3.21 bis in die Version 3.3 von Palsoft und den beiden in Gebrauch befindlichen Monitoren der Maschine ITT-2020 und ihrer Artverwandten bewährt, sofern diese Rechner mit 48 KByte RAM ausgerüstet waren. Ein geringeres Speichervolumen macht entsprechende Korrekturen einiger Vektoren erforderlich, Einsatz anderer Disketten-Betriebssysteme darüber hinaus Modifikation der DOS-Übergabe-Pointer. Soll PUT zusammen mit weiteren Hochsprachen-Übersetzern, etwa mit Integer Basic oder Pascal zusammenarbeiten, muß die „Umweg-Schaltung“ den dort gegebenen Erfordernissen ange-

paßt werden. Bei PUT-Betrieb an der Language-Karte sei empfohlen, die Karte nachzuladen, wenn man dort hausgemachte Routinen sitzen hatte. Sprachwechsel und MAXFILES-Kommandos mit Parametern größer 3 zerstören PUT ebenso wie Neubooten.

Abgesehen von diesen Einschränkungen, die für die praktischen Einsatzfälle beim Verfasser ohne jede Bedeutung waren, hat sich PUT als ein benutzerfreundliches und bequem zu handhabendes Mittel erwiesen, die stumpfsinnig hintereinandergedruckten Papierschlangen der Vergangenheit durch sauber paginierte und ordentlich formatierte Ausdrücke abzulösen.

Bleibt zum Schluß die Frage nach möglichen Erweiterungen von PUT zu beantworten. Wer glücklicher Besitzer einer Datum- und Uhrzeit-Karte von Art der „Mountain Hardware Apple Clock Card“ ist, kann mit mäßigem Aufwand PUT zusätzlich zum Ausdruck einer Zeitangabe jeweils am Seitenkopf veranlassen – bei Festlegung der Bufferbereiche ist hierfür ausdrücklich RAM-Platz freigelassen worden.

BF34.909B

```
BF34- A9 BF 85 74
BF3B- A9 34 85 73 20 44 90 AA
BF40- A0 D0 D5 D4 A0 C3 CF CE
BF4B- CE C5 C3 D4 C5 C4 AE AD
BF50- BD EA A9 6A 8D 53 AA A9
BF5B- BF 8D 54 AA A9 00 BD 70
BF60- 90 A9 01 8D 6E 90 BD 6F
BF6B- 90 60 8E 72 90 8C 73 90
BF70- 8D 71 90 20 11 90 C9 8C
BF7B- F0 23 C9 BD F0 11 A0 01
BF80- 8C 70 90 AD 71 90 AE 72
BF8B- 90 AC 73 90 4C 02 C1 20
BF90- BE BF 20 20 90 A0 00 8C
BF9B- 70 90 4C 83 BF AD 6E 90
BFA0- CD 76 90 F0 06 20 30 90
BFA8- 4C 9D BF 4C 83 BF AD 6E
BFBO- 90 CD 76 90 90 06 CD 77
BFBB- 90 B0 01 60 38 60 EE 6E
BFCC- 90 AD 6E 90 CD 74 90 D0
BFCE- 10 A9 01 8D 6E 90 AD 6F
BFDO- 90 F8 18 69 01 BD 6F 90
BFDB- DB AD 6E 90 CD 75 90 D0
BFEO- 03 20 E5 BF 60 A2 00 E0
BFEB- 24 F0 0A BD 78 90 20 02
BFEC- C1 E8 4C E7 BF AD 6F 90
BFEB- 4A 4A 4A 4A F0 05 09 B0
9000- 20 02 C1 AD 6F 90 29 0F
900B- 09 B0 20 02 C1 20 30 90
9010- 60 20 AE BF 90 06 20 30
901B- 90 4C 11 90 AD 71 90 60
9020- AC 70 90 D0 0A A9 A0 20
902B- 02 C1 A0 01 8C 70 90 60
9030- 20 BE BF 20 20 90 A9 BD
903B- 20 02 C1 A0 00 8C 70 90
9040- 60 20 58 FC 20 8E FD 68
904B- BD 53 90 68 BD 54 90 20
9050- 65 90 AD 00 00 C9 EA F0
905B- 06 20 ED FD 4C 4F 90 20
9060- 65 90 6C 53 90 EE 53 90
906B- D0 03 EE 54 90 60 EA EA
9070- EA EA EA EA 48 02 05 44
907B- A0 A0 A0 A0 A0 A0 A0 A0
9080- A0 A0 A0 A0 A0 A0 A0 A0
908B- A0 A0 A0 A0 A0 A0 A0 A0
9090- A0 A0 A0 A0 A0 AA A0 D0
909B- C1 C7 C5 A0
*
```

Bild 4. Objekt-Code des assemblierten Quelltextes aus Bild 3. Sofern auf Formatwahl nach Wunsch und die Einfügung des bis zu 27 Zeichen langen Seitenanfang-Textes verzichtet wird, reichen die 360 Byte dieses Listings allein

Literatur

- [1] Lübbert, Prof. William F.: „What's Where in the Apple“. MICRO Nr. 15, August 1979. Europa-Vertrieb: MSB Markdorf.
- [2] Little, Gary: „Paged Output for the Apple“. MICRO Nr. 29, Oktober 1980.
- [3] Willis, Jerry und Pol, Bernd: „Was der Mikrocomputer alles kann“. Vogel-Verlag, Würzburg.
- [4] Joepgen, Hans-Georg: „Der Euro-Apple – Erfahrungen mit dem ITT 2020“. FUNKSCHAU 1979, Heft 14.
- [5] Valentini, Johannes: „Ein (fast) ideales Speichermedium – das Disk-Operating-System des Euro-Apple“. FUNKSCHAU 1980, Heft 6.
- [6] Dougherty, Dr. William E.: „The Apple Monitor – Peeled“. Eigenverlag des Verfassers – 14349 San José Street, Mission Hills, California 91345, USA.
- [7] Sippl, Charles J.: „Lexikon der Mikro-Elektronik“. IWT-Verlag, München-Vaterstetten.
- [8] Mountain Hardware (Herausgeber): „Apple Clock Operating Manual“, Revision 2, July 1978. 300 Harvey West Blvd, Santa Cruz, California 95060, USA.
- [9] Joepgen, Hans-Georg: „Variablen-Wächter sorgt für mehr Programm-Transparenz“. ELEKTRO-NIK 1980, Heft 8.
- [10] Joepgen, Hans-Georg: „Bit-Flags ermöglichen elegante Basic-Wege“. FUNKSCHAU-Sonderheft „Programme für Kleincomputer und Taschenrechner“.

Rudolf Hofer

V.24- Ein-/Ausgabe für den Apple

Der Apple-II besitzt in seinem „Urzustand“ leider keine serielle Schnittstelle. Mit einem kurzen Maschinenprogramm läßt sich dieser Mangel aber ohne großen Hardware-Aufwand beseitigen.

Will man den Apple um Ein-/Ausgabe-Möglichkeiten erweitern, dann kommt man normalerweise nicht darum herum, sich eine zusätzliche Einsteckkarte zuzulegen. Eine serielle Ein-/Ausgabe-Schnittstelle kann man jedoch auch mit dem vorhandenen „Game I/O Connector“ realisieren. Die beiden Programme ZAUS und ZEIN sorgen dafür, daß die Daten über Anschluß 15 ausgegeben und über Anschluß 3 empfangen werden. Ein Zeichen wird nur dann gesendet, wenn der Busy-Eingang (2) auf L-Pegel liegt.

Umgekehrt geht der Busy-Ausgang immer dann auf L, wenn der Computer empfangsbereit ist. Verbindet man Daten- und Busy-Leitungen kreuzweise, dann ist auch eine Rechnerkopplung möglich. Um ZAUS zu aktivieren, sind die RAM-Zellen 36 und 37 (hex.) mit 00 03 zu laden (z. B. mit POKE-Befehlen oder vom Monitor aus). Ebenso wird ZEIN aktiviert, wenn man die RAM-Zellen 38 und 39 mit 49 03 (hex.) lädt. Die Übertragungsgeschwindigkeit ist auf 1200 Bd eingestellt. Geringere Werte

sind möglich, wenn man den Verzögerungswert 15 in Zeile 37 erhöht. Bei der Eingabe muß dann zusätzlich der Wert 10 in Zeile 63 so erhöht werden, daß sich eine Verzögerungszeit von einem halben Bit ergibt. Das Eingabeprogramm kann ein serielles Signal mit sieben oder acht Datenbits empfangen. Das achte Bit wird allerdings immer auf H gesetzt. Verhindern kann man das, indem man den ORA-Befehl in Zeile 82 durch EA EA ersetzt (Speicherzellen 380/381). Das Sendeprogramm schickt einen 7-Bit-Code. Es kann leicht auf 8 Bit eingestellt werden, indem man die Speicherstelle 30B auf 07 abändert. Um den Apple an eine V.24-Schnittstelle anschließen zu können, muß man noch die Pegel entsprechend anpassen [1]. Die in [2] beschriebene Schreibmaschinen-Schnittstelle kann direkt, ohne Zusatz-Hardware angesteuert werden.

Literatur

- [1] Klein, R.-D.: V.24-Interface. mc 1981, Heft 4, S. 34.
- [2] Hofer, R.: Interface für Typenrad-Schreibmaschine. „Das EMUF-Sonderheft“, Franzis-Verlag 1982.

```

0800      1  ;V24-AUS OHNE TIMER
0800      2  ;ANO=AUSGANG/GAME CONN 15
0800      3  ;PBO=BUSY--EINGANG/GAME CONN 2
0800      4  ;*****
0800      5  XTEM1 EQU $47A
0800      6  XTEMP EQU $77A
0800      7  PA EQU $C061
0800      8  ;*****
0300      9  ORG $300
0300 4B   10  ZAUS PHA
0301 4B   11  PHA
0302 BE7A04 12  STX XTEM1
0305 2C61C0 13  EB BIT PA
0308 30FB   14  BMI EB
030A A208   15  LDX #8 ;8 BIT
030C 203403 16  BEGA JSR WAIT ;7 BIT, WENN
030F A900   17  LDA #0 ;X=7
0311 BD58C0 18  STA $C058
0314 68     19  PLA
0315 4B     20  WIEDH PHA
0316 203403 21  JSR WAIT
0319 68     22  PLA
031A 203A03 23  JSR OUT2
031D 4A     24  LSR
031E 9000   25  BCC NOCRY
0320 CA     26  NOCRY DEX
0321 D0F2   27  BNE WIEDH
0323 A901   28  LDA #1
0325 4B     29  PHA
0326 203403 30  JSR WAIT
0329 68     31  PLA
032A 203A03 32  JSR OUT2
032D AE7A04 33  LDX XTEM1
0330 68     34  PLA
0331 4CF0FD 35  JMP $FDFO ;COUT
0334      36  ;*****
0334      37  WAIT LDA #15
0336 20A8FC 38  JSR $FCAB ;WAIT 1 BIT
0339 60     39  RTS
033A      40  ;*****
033A 4B     41  OUT2 PHA
033B 8C7A07 42  STY XTEMP
    
```

```

033E 2901   43  AND #%00000001
0340 AB     44  TAY
0341 9958C0 45  STA $C058,Y ;AUSG.
0344 AC7A07 46  LDY XTEMP NACH ANO
0347 68     47  PLA
0348 60     48  RTS
0349      49  ;*****
0349      50  PAG
0349      51  ;V24-EINGABE
0349      52  ;PB1=EINGANG/GAME CONN 3
0349      53  ;AN1=BUSY--AUSGANG/GAME CONN 14
0349      54  ;*****
0349      55  PB EQU $C062
0349      56  ;*****
0349 BE7A04 57  ZEIN STX XTEM1
034C BC7A07 58  STY XTEMP
034F BD5AC0 59  STA $C05A ;EB=0
0352 2C62C0 60  EMP BIT PB ;WARTEN
0355 30FB   61  BMI EMP ;AUF L
0357 A208   62  LDX #8
0359 A90A   63  LDA #10
035B 203603 64  JSR WAIT+2 ;1/2 BIT
035E 2C62C0 65  BIT PB ;START-
0361 30EF   66  BMI EMP ;BIT=0?
0363 A900   67  LDA #0
0365 4B     68  NBIT PHA
0366 203403 69  JSR WAIT
0369 AD62C0 70  LDA PB
036C 0A     71  ASL
036D 68     72  PLA
036E 6A     73  ROR
036F CA     74  DEX
0370 D0F3   75  BNE NBIT
0372 BD58C0 76  STA $C05B ;EB=1
0375 4B     77  PHA
0376 203403 78  JSR WAIT
0379 68     79  PLA
037A AE7A04 80  LDX XTEM1
037D AC7A07 81  LDY XTEMP
0380 0980   82  ORA ##80
0382 60     83  RTS
0382 60     84  END
    
```

Assembler-Listing
zur V.24-Ausgabe
über den Game-
Connector des
Apple-II

Andreas Lecreux

Apple lernt Kleinschrift

Einer der Hauptnachteile des Apple-II ist die fehlende Darstellung von Kleinbuchstaben. Mit wenig Aufwand läßt sich das aber beheben. Sie müssen dazu eine kleine Änderung auf der Tastaturdecodier-Platine vornehmen und einen anderen Zeichengenerator einsetzen. Nach dem Umbau funktioniert die Tastatur des Apple wie die einer Schreibmaschine – Großbuchstaben werden also mit gedrückter Shift-Taste eingegeben.

Um an die Decodier-Platine heranzukommen, muß man das Gehäuse des Apple von der Bodenplatte (sechs Rundkopfschrauben) abschrauben. Man sollte beide Teile allerdings erst voneinander lösen, wenn der Apple wieder mit dem Boden nach unten vor einem steht. Die Decodier-Platine ist mit der Hauptplatine über ein Flachkabel und einen DIL-Stecker verbunden. Ziehen Sie den Stecker vorsichtig heraus. Erst dann läßt sich der obere Teil mit Tastatur vollständig abnehmen.

Die Decodier-Platine steckt auf Plastiklaschen. Sie werden mit einer Flachzange

zusammengedrückt, während man die Platine vorsichtig herauszieht. Zwischen der Steckerleiste, die zur Tastatur führt, und dem Ctrl-Reset-Schalter befinden sich sechs Löcher mit Lötäugen, von denen je zwei über Leiterbahnen miteinander verbunden sind (Bild 1). Die beiden Leiterbahnen unterbricht man dort, wo sie in entgegengesetzten Dreiecken zusammenlaufen. Jetzt wird noch ein zweipoliger Umschalter (Bild 2) eingelötet, und der erste Teil des Umbaus ist abgeschlossen. Mit dem Umschalter kann man jederzeit vom ursprünglichen Zustand auf „Groß/Klein“ wechseln.

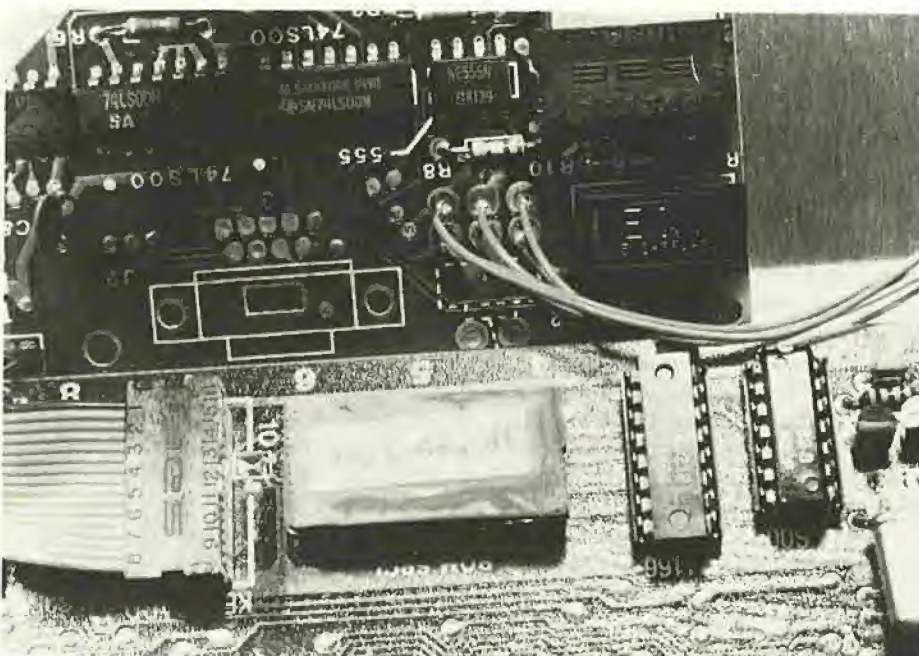


Bild 1. Der neu eingesetzte Zeichengenerator (von hinten auf den Apple gesehen) mit der ausgebauten Decodier-Platine

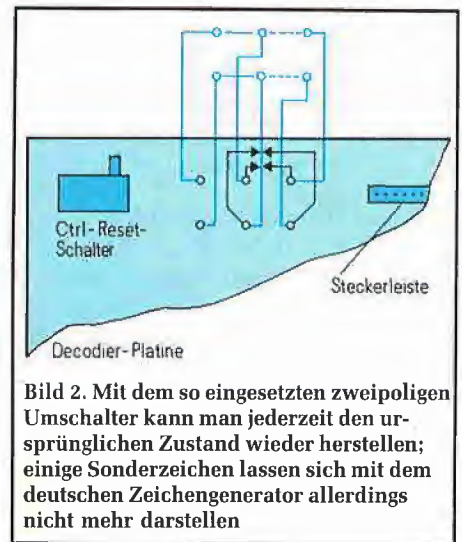


Bild 2. Mit dem so eingesetzten zweipoligen Umschalter kann man jederzeit den ursprünglichen Zustand wieder herstellen; einige Sonderzeichen lassen sich mit dem deutschen Zeichengenerator allerdings nicht mehr darstellen

Neuer Zeichengenerator notwendig

Damit man die Kleinbuchstaben auch auf den Bildschirm bekommt, ist ein neuer Zeichengenerator erforderlich. Dieses Bauteil sitzt direkt unter der Decodierplatine auf der Hauptleiterplatte des Apple (links neben dem schon erwähnten DIL-Stecker). Einfach ist die Sache, wenn Sie einen neueren Apple besitzen, der schon mit dem ROM 2316 ausgerüstet ist. Dieser Baustein läßt sich nämlich durch das Standard-EPROM 2716 ersetzen, das selbst programmiert werden kann. Lediglich die Anschlüsse 21 und 24 sind beim 2716 zu verbinden. Etwas komplizierter wird es, wenn als Zeichengenerator noch das ROM 2513 verwendet wird. Um es durch ein 2716 ersetzen zu können, muß man sich einen kleiner Adapter bauen. Bild 3 zeigt, welche Anschlüsse des EPROMs mit den einzelnen Anschlüssen der ROM-Fassung zu verbinden sind.

In Basic keine Änderung

Bei normaler Eingabe in Basic kann man trotz der Modifizierung zunächst keine Veränderung feststellen. Das liegt daran, daß die Eingaberoutine des Monitors alle Buchstaben in Großbuchstaben umwandelt. Da man Groß-/Kleinschreibung ohnehin meist nur bei Textverarbeitungsprogrammen haben will (die fast immer eine eigene Eingaberoutine besitzen), soll hier nur angedeutet werden, wie man auch in Basic zu Kleinbuchstaben kommt: Entweder man schreibt sich eine eigene Eingaberoutine und lenkt den Eingabe-Vektor in der Zero-Page um, oder man verändert das Monitorprogramm (ohne Softcard nur mit neuem

Apple-II-Karte

Von der Hannoveraner Firma Springmann wird eine EPROM-Programmierschaltung für den Apple-II angeboten, die sich für alle gängigen EPROM-Typen (einschließlich 27128) eignet. Sie wird für den Apple-Slot 2 konfiguriert geliefert; die nötige Software befindet sich auf der zugehörigen Diskette.

Leider ist es erforderlich, bei jedem EPROM-Typenwechsel eine Anzahl von Drahtbrücken umzustecken. Dabei ist allerdings kein Lötkolben erforderlich, weil für diesen Zweck ein IC-Sockel vorhanden ist. Eine weitere Steckbrücke dient zum Umschalten der Programmierspannung von 26 V auf 21 V (die Typen 2764 und 27128 werden mit nur noch 21 V programmiert).

Ein weiterer kleiner Haken ist, daß das (programmtechnisch sonst gut unterstützte) Anfertigen von EPROM-Serien etwas umständlich ist. Zum einen befindet sich der „Nullkraft“-Textool-EPROM-Sockel direkt auf der Platine, so daß sich je nach Bestückung des Apple mit anderen Karten das Hineinstecken und Herausnehmen von EPROMs u. U. nur durch Herausziehen der ganzen Karte realisieren läßt. Ein per Kabel absetzbarer Sockel wäre sicher besser.

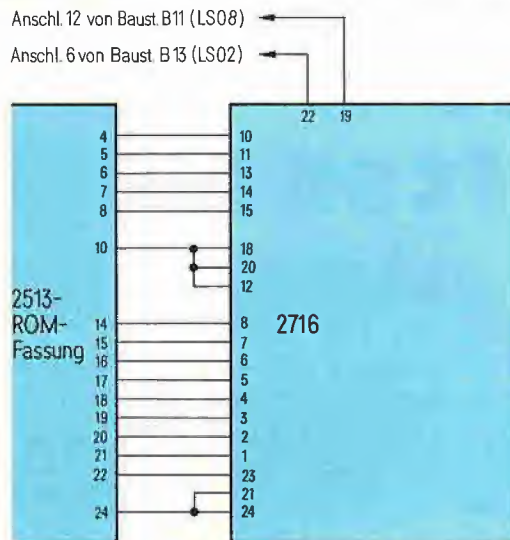
Zum anderen ist die Software zum „Burner Plus“ so aufgebaut, daß man gezwungen ist, jedesmal vor dem Programmieren eines neuen EPROMs den entsprechenden Speicherbereich neu von Diskette zu lesen, nicht zuletzt deshalb, weil die mitgelieferte Dokumentation (mit Schaltbild!) davon ausgeht, daß der Apple jeweils während des EPROM-Wechsels ausgeschaltet wird und somit der RAM-Inhalt verloren geht.

Die Praxis hat jedoch gezeigt, daß diese Vorsichtsmaßnahme nicht unbedingt erforderlich ist. Das Aus- und Einschalten des Computers sowie das Neuladen von Programm und EPROM-Speicherbereich könnte man sich also durch Ändern der Software auch sparen. Gerade dann empfiehlt es sich allerdings dringend, den EPROM-Sockel über ein Flachkabel herauszuführen, um nicht bei eingeschaltetem Gerät im Innern herumfuchswerken zu müssen.

Lobenswerterweise macht das Begleitmaterial auch Angaben darüber, wie man CP/M-Dateien in EPROMs übernehmen kann. Damit eignet sich der Apple-II als universelles Entwicklungssystem für die Prozessoren 6502 und Z80.

Herwig Feichtinger

Bild 3. Anstelle des „alten“ Zeichengenerators 2513 kann auch ein EPROM verwendet werden, wenn man (am besten mit Hilfe eines Adapters) die Anschlüsse wie angegeben verbindet



EPROM möglich). Beim Autostart-Monitor sind drei Befehle ab Adresse FD7E für die Klein/Groß-Umwandlung verantwortlich; sie sind durch sechs NOP-Befehle (FD7E...FD83) zu ersetzen.

Ein weiterer Weg steht noch offen – Sie sollten ihn wählen, um zu testen, ob der Umbau erfolgreich war: Der Befehl

X=PEEK (-16384)

weist der Variablen X unmittelbar den ASCII-Wert der gedrückten Taste zu. Mit PRINT CHR\$(X) kann dann auch jeder Kleinbuchstabe dargestellt werden.

Ein Zeichengenerator mit Umlauten und Unterlängen kann vom Franzis-Software-Service bezogen werden.

Mehr Leistung für Apple-II

Orange Speed heißt ein neues Produkt, das den Apple II insbesondere für technisch-wissenschaftliche Anwendungen auf Vordermann bringt. Es besteht aus einer Einsteckkarte, die mit dem Arithmetik-Prozessor Am9511A ausgestattet ist, einem ROM-Adapter, zwei Disketten und einem umfangreichen Handbuch. Der Arithmetik-Prozessor kann von einer beliebigen Sprache aus angesprochen werden. Er erledigt Rechenoperationen bis zu 100mal schneller als der 6502. Neben Ganzzahlen (± 32768 oder ± 2147483647) verarbeitet er auch Gleitkommazahlen ($\pm 0,9223367 \cdot 10^{19}$). Die möglichen Rechenoperationen reichen von den vier Grundrechenarten bis zum Wurzelziehen, zu trigonometrischen und Exponentialfunktionen. Daneben sind zahlreiche Befehle für Stackmanipulationen und Formatumwandlungen vorgesehen.

Der eigentliche Clou von Orange Speed aber ist die Software. Sie besteht im wesentlichen aus einem Compiler für „Metalanguage“ – einer stackorientierten Sprache, die nach Art mancher Taschenrechner in der umgekehrten polnischen Notation arbeitet. Diese Sprache kennt als einziges Element den Befehl. Ausge-

hend von einem Grundbefehlssatz „baut“ sich der Programmierer eigene Befehle, die dem Wortschatz angefügt werden. Ein Programm besteht schließlich aus einem einzigen Befehl, der die gestellte Aufgabe ausführt. Nachteil: Der Programmierer ist für die Verwaltung des Stack zu jeder Zeit selbst verantwortlich und wird vom System kaum auf Fehler aufmerksam gemacht. Leute, die es gewöhnt sind, Basic-Programme zu erstellen, dürften sich anfangs damit sehr schwer tun. Dies sind allerdings nicht die typischen Anwender. Besonders geeignet ist die Sprache zusammen mit dem Zusatzprozessor für Grafikanwendungen, bei denen viel berechnet werden muß. Wer schon einmal am Bildschirm erlebt hat, wie mühsam Punkt für Punkt an eine Kurve angefügt wird, der weiß den erheblichen Geschwindigkeitszuwachs zu schätzen. Befehle zum Erstellen von Polygonzügen und zum Einfügen von Text in die Grafikseite erhöhen den Komfort. Das ganze System ist kompatibel zu DOS 3.2/3.3, Applesoft, CP/M und Pascal. Hervorheben muß man die ausgezeichnete deutsche Dokumentation.

Rudolf Hofer

Klaus-Dieter Schumacher

Apple-II: Diskettenkapazität preiswert erhöht

Sobald umfangreichere Anwendungen mit dem Apple-II geplant sind, stößt man immer wieder auf das nach Ansicht des Verfassers gravierendste Problem des zu geringen externen Speicherplatzes. Die vom Hersteller angebotenen Disk-II-Laufwerke bieten sparsame 140 KByte (formatiert) je Laufwerk.

Als Alternative findet man auf dem deutschen und amerikanischen Markt von verschiedenen Anbietern Lösungen auf der Basis von 8-Zoll-Laufwerken oder Harddisks. Diesen Lösungen ist allen gemeinsam, daß sie für den Benutzer sehr teuer sind (ab ca. 5000 DM). Weiter bestehen meist Probleme mit Software-Paketen wie Visicalc oder Inkompatibilitäten mit speziellen Interfacekarten. Zusätzlich wird die Kommunikation zu anderen Benutzern des Apple-II erschwert, da z. B. ein Datenaustausch mit Besitzern von 5,25-Zoll-Laufwerken nicht mehr möglich ist.

Eine weitere Alternative, die Doppeldiskettenstation der Firma Commodore über ein IEC-Bus-Interface anschließbar, scheidet ebenfalls aus Kosten- und Inkompatibilitätsgründen aus. Ähnlich sind die neuen Doppeldiskettenstationen der Firma Apple für das Modell III zu bewerten.

Auf dem Markt werden nun jedoch seit einigen Wochen die Slimline-Laufwerke der Firma TEAC (Typ FD55) angeboten. Sie haben bei halber Bauhöhe die gleichen Abmessungen wie die Disk-II-Lauf-

werke der Firma Apple. Neben den 40- und 80-Spur-Versionen gibt es nun auch solche mit zwei Schreib-/Leseköpfen, einen für die Oberseite und einen für die Unterseite der Diskette.

Das am Apple-II unter Pascal II.1, IV.0, und CP/M betriebene TEAC-Laufwerk FD55F hat bei 2×80 Spuren eine Kapazität von 640 KByte (formatiert). An einen Controller, z. B. der Firma Ehring-Electronic, lassen sich direkt zwei solche Laufwerke anschließen. Bei weiteren Firmen auf dem deutschen und amerikanischen Markt sind Controller erhältlich, an die bis zu vier Laufwerke angeschlossen werden können. Hierbei sind jedoch fast immer Änderungen an der Hardware für den „Sideselect“ der Laufwerke notwendig.

Will man am Betriebssystem keine zu großen Modifikationen vornehmen, sind unter CP/M und Pascal maximal 3×2 Laufwerke anschließbar. Dies bedeutet eine externe Gesamtspeicherkapazität von etwa 4 MByte formatiert ($6 \times 640 \times 1024$ Byte). Dies ist erfahrungsgemäß für sehr viele Applikationen mehr als ausreichend. Bereits mit zwei Laufwerken hat man keine Platzprobleme mehr bei Programmen wie Wordstar, PL/1 oder MT+.

Eine bereits vorliegende Betriebssystemmodifikation unter Apple-Pascal 1.1, die alle 160 Spuren wie ein Laufwerk anspricht, zeigte bisher keine Probleme. Ebenso ist nach bisherigen Erfahrungen ein Anpassen an die UCSD-Pascal-Version IV.0 unter Verwendung des „Instal-

lation Guide“ unproblematisch.

Eine entsprechende Modifikation für den Betrieb unter CP/M existiert noch nicht. Zur Zeit stehen je Seite 306 KByte zur freien Verfügung, wenn beide Seiten als getrennte Laufwerke gefahren werden.

Bis eine Betriebssystemmodifikation für den Betrieb des FS55F unter CP/M erarbeitet ist, kann man sich mit der im Bild dargestellten Schaltung behelfen, die zeigt, wie man Vorder- und Rückseite als erstes und zweites Laufwerk anspricht. Die Laufwerke sind sauber verarbeitet. Sie sind mit einem Motor ausgerüstet, der die Diskette unmittelbar, d. h. ohne Riemen, betreibt. Einschließlich Controller, Kabel und Stecker sind zwei Laufwerke ohne Gehäuse für ca. 2200 DM erhältlich.

Ogleich bei diesen Fremdlaufwerken erstmalig die Möglichkeit gegeben ist, sie wegen ihrer geringen Stromaufnahme direkt über das Netzteil des Rechners zu versorgen, empfiehlt sich ein zusätzliches oder verstärktes Netzteil. Der Netztrafo sollte jedoch wegen möglicher Streuung nicht neben oder hinter dem Laufwerk montiert sein.

Geschützte Software, z. B. das Programm Visicalc, ist ebenfalls ohne Probleme sofort verwendbar, da die Laufwerke durch das Umsetzen einer Brücke (Strap to designate track density) auf den 35-Spur-Betrieb der Apple-II-Laufwerke umgestellt werden können.

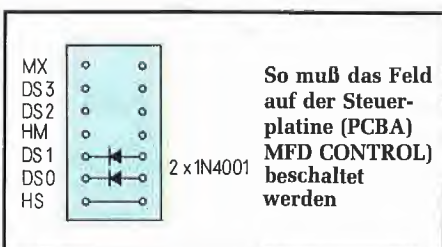
Will man die bisher in Apple- oder vergleichbaren Laufwerken beschriebenen Rückseiten lesen und beschreiben, so kann das Signal des Fototransistors für das Indexloch durch eine kleine Modifikation abgeschaltet werden (Achtung: Garantieverlust!).

Geschützte Software, die auf die Halbspuren der Disk-II-Laufwerke zugreift (z. B. DB-Master) kann noch nicht direkt verwendet werden.

Bei Anschluß mehrerer Laufwerke ist es empfehlenswert, das Motor-on-Signal über den „Drive-Select“ zu führen. Man reduziert die Leistungsaufnahme des nicht angesprochenen Laufwerkes auf ca. 2,6 W, während das benutzte Laufwerk etwa 5,5 W benötigt.

Literatur

- [1] Stark, Peter A.: Firm up your Floppy with 800k. Microcomputing, Aug. 1981, Seite 36.
- [2] TEAC FD-55 Mini flexible disk drive specification. UCSD p-System. Version IV. Installation Guide. Softtech, 1982. Vertrieb: NBN Elektronik GmbH, 8036 Herrsching; Ehring Elektronik, 4100 Duisburg.



Klaus Rüdiger Hase

Universal-Schnittstelle für Apple-II

Dieser Artikel beschreibt eine universelle Schnittstelle für den Apple-II, die mit den Betriebssystemen DOS 3.3, Apple-Pascal (Vers. 1.0 und 1.1) und CP/M 2.2 einsetzbar ist. Für die Software wird ein „Kochrezept“ in Form eines Rahmenprogramms angegeben, das vom Benutzer nur noch um ein anwenderspezifisches Treiberprogramm ergänzt werden muß. Universell ist die Schnittstelle deshalb, weil sich der Programmspeicher wahlweise mit einem 2-KByte-RAM oder -EPROM ausstatten läßt und der VIA-Baustein 6522 serielle und parallele Übertragung ermöglicht.

Das Schnittstellenkonzept des Apple-II

Der Apple-II zeichnet sich besonders durch seine flexible Lösung des Schnittstellenproblems aus, da er mit verschiedenartigen Interfaces für Drucker, Plotter, Modems usw. ausgestattet werden kann. Für diesen Zweck hat er auf seiner Leiterplatte sieben Steckplätze (Slots), die schon eine Adreßauswahl für Schnittstellen-Bausteine und Speicher (für die Treiberprogramme) vornehmen.

Ein kleiner Schönheitsfehler macht aber auch hier dem Schnittstellenprogrammierer das Leben nicht ganz leicht. Zwar findet sich in den Unterlagen [1] eine Reihe von Hinweisen, wie man eine Schnittstelle aufbauen und programmieren kann, dabei mag es auch gelingen, einen alten Fernschreiber oder eine elektrische „Hammerschreibmaschine“ unter DOS 3.3 für Basic oder den Monitor lauffähig zu bekommen; groß wird aber sicherlich die Enttäuschung sein, wenn sich unter Pascal oder CP/M gar nichts mehr rührt.

Da hilft auch der Hinweis in den CP/M-Unterlagen nicht weiter, daß fast alle unter Pascal anschließbaren Schnittstellen auch mit CP/M verwendbar seien, man aber nach den Erfordernissen für die Treibersoftware in den Pascal-Handbüchern vergeblich sucht. Diese Enthalt-

samkeit der Firma Apple ist um so unverständlicher, als doch die zur Grundversion gelieferten Handbücher recht ausführlich und vollständig sind.

Einzig Licht in dieses Dunkel bringt Barry Haynes mit seinem „ATTACH-BIOS document for Apple II Pascal 1.1“, das allerdings den unerfahrenen Programmierer gleich im vierten Satz mit dem Hinweis abschreckt: „This document is intended for more advanced users who already know amount about I/O devices... It is not intended to be a simple step by step description of how to write your first device driver...“

Deshalb soll an dieser Stelle ein „Kochrezept“ vorgestellt werden, das es praktisch jedermann ermöglichen wird, eigene Peripherie anzupassen. Die Software ist demnach nur ein Rahmenprogramm, das der Benutzer um sein eigenes Treiberprogramm ergänzen muß. Das Rahmenprogramm gestattet die Ausgabe und Eingabe einzelner ASCII-Zeichen, ist also zeichenorientiert, im Gegensatz zu den unter Pascal auch möglichen blockorientierten Schnittstellen.

Im Apple-Pascal hat jede Schnittstelle ihren festen Platz

Während der Basic-Interpreter des Apple-II unterschiedslos jeden Slot mit PR#n oder IN#n (n = Slot-Nr.) bedient, gilt für

Pascal und CP/M eine feste Zuordnung, die nur durch Eingriffe in das BIOS (Basic I/O Subsystem) verändert werden kann. Dies kann aber nur dem (siehe oben) „more advanced user“ empfohlen werden, der sich mit [2] hinreichend vertraut gemacht hat.

Für das Apple-Pascal gelten folgende Zuordnungen:

Slot 0:	Language-Karte kein I/O
Slot 1:	Printer: nur Zeichen-Ausgabe
Slot 2:	Remin., Remout: Zeichen Ein-/Ausgabe
Slot 3:	Console: Zeichen Ein-/Ausgabe
Slot 4:	Diskette: blockorientiert
Slot 5:	Diskette: blockorientiert
Slot 6:	Bootdisk: blockorientiert
Slot 7:	PAL-Karte/Z80-Softcard

Diese Zuordnung gilt mit etwas anderen Bezeichnungen aber prinzipiell gleichen Funktionen auch für CP/M 2.2 von Microsoft. Wir werden unsere Aufmerksamkeit nun auf die drei zeichenorientierten Schnittstellen richten, also auf die Schnittstellen in Slot 1 bis 3.

Reservierte Adreßbereiche für Apple-Slots

Für Ein-/Ausgabe-Schnittstellen der verschiedensten Art ist der Adreßbereich von \$C000 bis \$CFFF reserviert, der sich wiederum in folgende Bereiche unterteilt:

1. Der Bereich von \$C000...\$C07F gehört den internen I/O-Registern (Tastatur, Paddles, Bildschirm usw.).
2. Im Bereich von \$C080...\$C0FF können jeweils 16 (\$10) Adressen mit dem Signal Device-Select decodiert werden, um damit Schnittstellenbausteine (VIA, ACIA usw.) anzusteuern. Die Zuordnung erfolgt so, daß mit der Basisadresse \$C080 und einem Index von \$n0 im Y-Register die Schnittstelle erreicht werden kann, wobei „n“ wieder die Slot-Nr. angibt.
3. Der Bereich von \$C100...\$C7FF wird mit je einer Page (256 Byte, \$Cn00...\$CnFF) den einzelnen Slots mit dem Signal I/O-Select zugewiesen. Die Basisadressen \$Cn00 sind die Einsprungpunkte für die Schnittstel-

C115	C168	PHA	C168	OUTBA1	A524	LDA CH	;Pruefe ob Pointer in \$24 dem
C116	C168	TYA	C168	OUTBA1	BD3804	CMP CHRZAL,X	;Zeichenzaehler voraussetzt
C117	C168	PHA	C168	OUTBA1	BD00D	BCS INCSPC	;z.B. bei TAB
C118	C168	ADFCF LDA \$CEFF	C168	OUTBA1	C911	CMP #811	;falls groesser als 17, dann
C119	C168	ADFCF JSR \$FF58	C168	OUTBA1	B009	BCS INCSPC	;zusätzliche Leerzeichen
C120	C168	BA	C168	OUTBA1	09F0	ORA #8F0	;setze hoeherwert. Bits
C121	C168	BD0001 LDA \$0100,X	C168	OUTBA1	3D3804	AND CHRZAL,X	;mit aktuellem Zaehlerstand
C122	C168	8DF807 STA SLOTR	C168	OUTBA1	6524	ADC CH	;Korrektur (carry = 0 !)
C123	C168	0A ASL A	C168	OUTBA1	8524	STA CH	
C124	C168	0A ASL A	C168	OUTBA1	BD3804	LDA CHRZAL,X	
C125	C168	0A ASL A	C168	OUTBA1	C524	CMP CH	;sind beide Zaehler gleich ?
C126	C168	0A ASL A	C168	OUTBA1	68	PLA	;weiter, wenn kein TAB
C127	C168	0A ASL A	C168	OUTBA1	B003	BCS NOSPC	
C128	C168	0A ASL A	C168	OUTBA1	48	PHA	;Auffuellen mit Leerzeichen
C129	C168	8DF806 STA SLOTOF	C168	OUTBA1	A9A0	LDA #8A0	;kein Contrl-Zeichen ?
C130	C168	A8	C168	OUTBA1	2C58FF	BIT \$FF58	;dann kein Increment
C131	C168	68 PLA	C168	OUTBA1	F003	BEQ NOINC	;nur bei druckbaren Zeichen
C132	C168	68 PLA	C168	OUTBA1	FE3804	INC CHRZAL,X	
C133	C168	68 PLA	C168	OUTBA1	08	PHP	
C134	C168	68 PLA	C168	OUTBA1	BD3804	LDA CHRZAL,X	
C135	C168	68 PLA	C168	OUTBA1	C524	CMP CH	
C136	C168	68 PLA	C168	OUTBA1	68	PLA	
C137	C168	68 PLA	C168	OUTBA1	B003	BCS NOSPC	
C138	C168	68 PLA	C168	OUTBA1	48	PHA	
C139	C168	68 PLA	C168	OUTBA1	A9A0	LDA #8A0	
C140	C168	68 PLA	C168	OUTBA1	2C58FF	BIT \$FF58	
C141	C168	68 PLA	C168	OUTBA1	F003	BEQ NOINC	
C142	C168	68 PLA	C168	OUTBA1	FE3804	INC CHRZAL,X	
C143	C168	68 PLA	C168	OUTBA1	08	PHP	
C144	C168	68 PLA	C168	OUTBA1	BD3804	LDA CHRZAL,X	
C145	C168	68 PLA	C168	OUTBA1	C524	CMP CH	
C146	C168	68 PLA	C168	OUTBA1	68	PLA	
C147	C168	68 PLA	C168	OUTBA1	B003	BCS NOSPC	
C148	C168	68 PLA	C168	OUTBA1	48	PHA	
C149	C168	68 PLA	C168	OUTBA1	A9A0	LDA #8A0	
C150	C168	68 PLA	C168	OUTBA1	2C58FF	BIT \$FF58	
C151	C168	68 PLA	C168	OUTBA1	F003	BEQ NOINC	
C152	C168	68 PLA	C168	OUTBA1	FE3804	INC CHRZAL,X	
C153	C168	68 PLA	C168	OUTBA1	08	PHP	
C154	C168	68 PLA	C168	OUTBA1	BD3804	LDA CHRZAL,X	
C155	C168	68 PLA	C168	OUTBA1	C524	CMP CH	
C156	C168	68 PLA	C168	OUTBA1	68	PLA	
C157	C168	68 PLA	C168	OUTBA1	B003	BCS NOSPC	
C158	C168	68 PLA	C168	OUTBA1	48	PHA	
C159	C168	68 PLA	C168	OUTBA1	A9A0	LDA #8A0	
C160	C168	68 PLA	C168	OUTBA1	2C58FF	BIT \$FF58	
C161	C168	68 PLA	C168	OUTBA1	F003	BEQ NOINC	
C162	C168	68 PLA	C168	OUTBA1	FE3804	INC CHRZAL,X	
C163	C168	68 PLA	C168	OUTBA1	08	PHP	
C164	C168	68 PLA	C168	OUTBA1	BD3804	LDA CHRZAL,X	
C165	C168	68 PLA	C168	OUTBA1	C524	CMP CH	
C166	C168	68 PLA	C168	OUTBA1	68	PLA	
C167	C168	68 PLA	C168	OUTBA1	B003	BCS NOSPC	
C168	C168	68 PLA	C168	OUTBA1	48	PHA	

[illegible]

Bild 1. Listing des universellen Ein-/Ausgabe-Programms: Die Adressen des EPROMs (siehe Bild 2) sind so decodiert, daß der von \$C100...\$C1D3 assemblierte Code ab EPROM-Adresse \$700 abgelegt wird; der im Listing ab \$C800 wiedergegebene Bereich beginnt im EPROM ab \$000

Das Apple-Sonderheft

len-Initialisierung unter Basic mit PR#n und IN#n. Für viele Anwendungen wird dieser ¼ KByte große Bereich ausreichend sein.

- Der Rest von \$C800...\$CFFF ist zusammenhängend und kann von jeder Schnittstelle (Slot 1 bis Slot 7) über das Signal I/O-Strobe wahlweise in Anspruch genommen werden. Damit aber kein Buskonflikt bei Verwendung mehrerer Schnittstellen verursacht wird, ist ein elektronischer Schalter erforderlich, der diesen Bereich für alle übrigen Mitbenutzer abschaltet, bevor der eigene Speicher benutzt werden kann. Dies wird zwangsweise dadurch erreicht, daß der Zugriff nur über ein Enable-Flipflop möglich ist, das durch Ansprechen einer beliebigen Slotadresse (\$Cn00...\$CnFF für Slot n) gesetzt wird und durch Ansprechen von \$CFFF zurückgesetzt werden kann. Daher wird während des Bootvorganges diese Rücksetzadresse angesprochen, bevor irgendein Zeichen eine Schnittstelle passiert.

Die nun folgenden Speicherplätze können oder müssen von den Schnittstellenunterprogrammen benutzt werden, liegen aber nicht im genannten I/O-Bereich:

- Für die Ablage bestimmter Parameter (Baudrate, Zeichenzahl/Zeile) steht ein nicht zusammenhängendes Scratchpad-RAM im Bereich des

Text- und Low-Resolution-Grafik-Bildschirmspeichers zwischen \$478 und \$7FF zur Verfügung. Einzelheiten sind in [1] und im Listing (Bild 1) zu finden. Die Basisadressen \$478, \$4F8...\$7F8 sind für alle Schnittstellen zugänglich, während die Speicher, die mit der Basisadresse + \$0n erreicht werden können, nur den jeweiligen Slots mit der Nr. n vorbehalten sind.

Feste Vereinbarungen sind hierbei:

\$7F8 enthält die Slot-Nr. der aktiven Schnittstelle in der Form \$Cn;
\$5F8 trägt unter DOS die Slot-Nr. der Boot-Diskette (also nicht im Programm verändern!)

und speziell unter Apple-Pascal:

\$6F8 enthält die Slot-Nr. der aktuellen Schnittstelle mit \$n0.

\$678+\$0n = \$5B8+\$Cn enthält das zu übergebende ASCII-Zeichen, das bei der Ausgabe auch im Akku steht (Serial Card).

Unter Pascal steht überdies noch der Zeropage-Bereich von \$0...\$35 als Scratchpad (engl. Notizblock) zur Verfügung. Dieser Bereich wird aber nach Verlassen der Interfacerroutine durch das Betriebssystem selbst wieder „bombardiert“.

Auf seine Benutzung sollte man allerdings unter DOS 3.3 verzichten, da fast die gesamte Zeropage belegt ist.

Hardwareaufbau mit kleinen Tricks

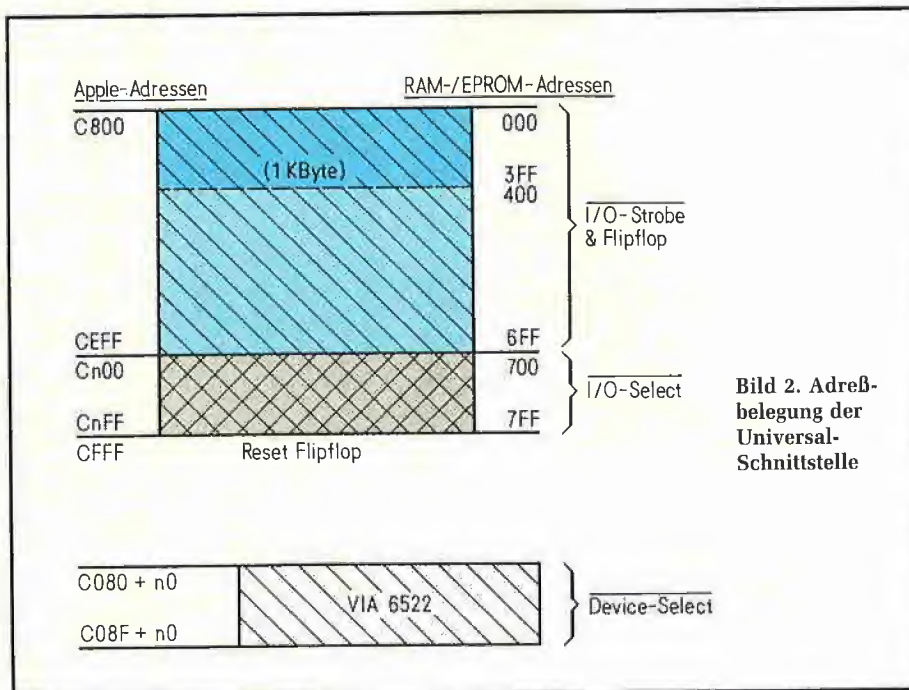
Dem Benutzer stehen also pro Slot insgesamt 2¼ KByte für Treiberprogramme zur Verfügung. Da eine preisgünstige Lösung angestrebt wird, soll nur ein 2-K-Speicher (wahlweise RAM oder EPROM) eingesetzt werden, bei dem der Basisbereich \$Cn00...\$CnFF an das obere Ende des gemeinsamen I/O-Bereiches von \$C800...\$CFFF gelegt und überlappend adressiert wird. Die Adreßbelegung ist genauer im Bild 2 dargestellt.

Für „endgültige“ Lösungen wird man als Speichermedium ein EPROM vom Typ 2716 bevorzugen, will man aber mit derselben Hardware verschiedene Aufgaben erledigen, sollte man auch hier flexibel sein und das fast anschußkompatible 2-K-RAM (z. B. Hitachi M58725P) verwenden, zumal man das Treiberprogramm nur einmal laden muß. Der Inhalt dieses RAM-Bereiches bleibt auch dann erhalten, wenn man von einem Betriebssystem auf ein anderes überwechselt bzw. Reset betätigt.

Die Vorzüge eines RAMs lernt man sehr schnell schätzen, wenn man einen größeren Pufferspeicher benötigt. Das Auswahl-Flipflop schützt auch dann vor Überschreiben, wenn z. B. eine 80-Zeichen-Karte mit eigenem Bildauffrischspeicher in \$CC00...\$CDDF (z. B. Videx) betrieben wird. Man darf sich allerdings bei der Initialisierung der Schnittstelle nicht über die wundersamen Muster auf dem Bildschirm wundern, die sich aber mit einem Ctrl-L schnell wieder löschen lassen.

Die Schaltung selbst ist in Bild 3 dargestellt. Ein kleiner Schönheitsfehler der Decodierungssignale (Dev.-Select) soll nicht verschwiegen werden: Die Apple-Entwickler haben offenbar keine Rücksicht auf das komplexe Timing der 65XX-Schnittstellen-Bausteine genommen. So kommt es wohl, daß das Enable-Signal, das aus dem Φ1-Signal durch Invertierung gewonnen werden muß, fast gleichzeitig mit dem als Chip-Select verwendbaren Device-Select anliegt, obwohl nach den Herstellerangaben (VIA 6522, PIA 6520, ACIA 6551 usw.) Chip-Select mindestens 100 ns vor dem Enable-Signal gültige Pegel annehmen muß.

Berücksichtigt man dies nicht, kann man die Schnittstellen-Bausteine überhaupt nicht adressieren. Abhilfe kann hier eine Verzögerung des invertierten Φ1-Signals werden, wobei die Signal-



dauer ($\Phi = \text{high}$) für 1-MHz-Bausteine 350 ns nicht unterschreiten darf. In Zweifelsfällen hilft ein Blick auf das Oszilloskop, um die Zugriffszeiten nicht zu arg zu strapazieren. Der gesamte Hardwareaufbau erfolgt am einfachsten auf einer Prototypenkarte, die überall erhältlich ist und zudem noch genügend Platz für weitere Schnittstellen-Bausteine läßt.

Ein Rahmenprogramm für alle Fälle

Das Interfaceprogramm ist unter Basic für alle Slots ohne Änderung lauffähig. Das bedeutet, daß es mit jeder Basisadresse \$Cn00 mit $n = 1...7$ gestartet werden kann und daher im Bereich der I/O-Page \$Cn00...\$CnFF nur relative Sprünge erlaubt sind. Dies ist bei Programmänderungen unbedingt zu beachten.

Die solchermaßen erreichte Slotunabhängigkeit erfordert, daß das Programm die Lage seiner I/O-Schnittstellen-Bausteine kennt. Der Basic-Interpreter übergibt hierzu keine Parameter, weshalb der Offset \$n0 zur VIA-Basisadresse \$C080 (z. B. \$10 für \$C090 = \$C080 + \$10 für Slot-Nr. 1) vom Programm selbst gefunden werden muß. Dies ist einfach, wenn man einen unveränderlichen Speicher-

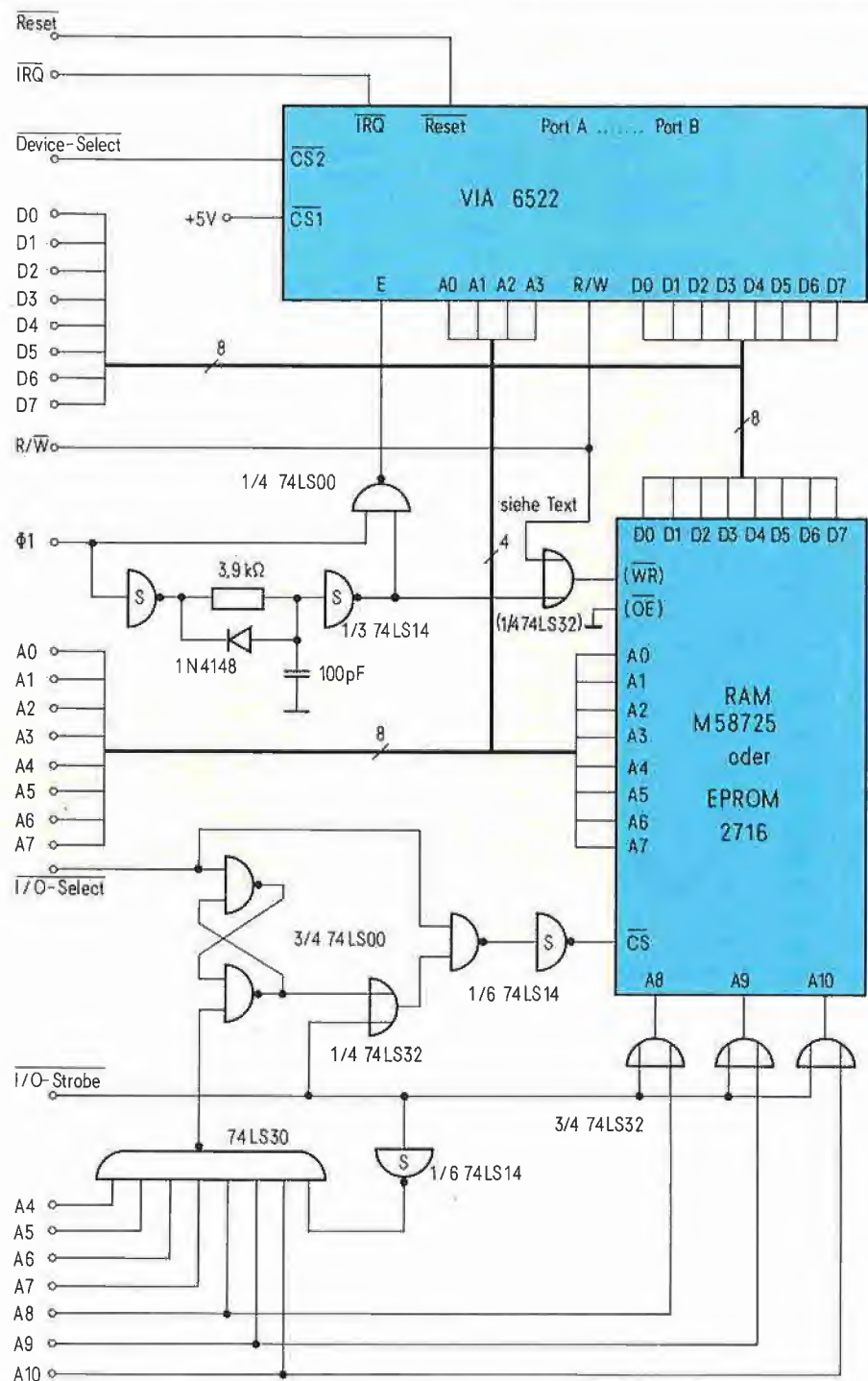


Bild 3. Schaltung der Universal-Schnittstelle

platz kennt, der \$60 (RTS) enthält. Springt man diese Stelle mit JSR an, so passiert nichts weiter, als daß die Rücksprungadresse auf den Stack gelegt wird. Der Stackpointer zeigt nach dieser Operation noch auf das höherwertige Byte der Rücksprungadresse, also der Adresse, in deren Bereich sich das aufrufende Programm gerade befindet. In unserem Fall wäre das aber gerade \$Cn (siehe Bild 1).

Mit der einfachen Folge:

```
$CnXX:JSR $FF58 ; Sprung nach
                    festem RTS
TSX ; Stackpointer
    ; in den Index (X)
LDA $100,X; $Cn in den
    Akku laden
```

kennt man diese Slotnummer und hat sie im Akku stehen. Für die weitere Verwendung wird nun in [1] empfohlen, die so codierte Slot-Nr. im Scratchpad-Register \$7F8 abzulegen und für die indizierte Adressierung der anderen Speicher im Scratchpad-Bereich dem Y-Register zu übergeben.

Um die I/O-Schnittstellenregister zu erreichen, muß der Index noch mit viermaligem ASL A in die Form \$n0 überführt werden. Der solchermaßen aufbereitete Index wird in das X-Register übertragen. Diese Belegung der Indexregister wird auch von den Betriebssystemen Apple-Pascal und Apple-CP/M 2.2 beibehalten.

Den Manipulationen mit der Slot-Nr. folgt ein kleines Initialisierungsprogramm (INITS), das die Ein-/Ausgabevektoren des Basic-Interpreters in \$36/\$37 (Ausgabe) und \$38/\$39 (Eingabe) so verändert, daß das Programm mit unterschiedlichen Einsprungvektoren für beide Verarbeitungsrichtungen ausgestattet wird. An dieser Stelle kann auch eine Unterscheidung für die Schnittstellen-Initialisierung vorgenommen werden. Für jedes weitere auszugebende Zeichen wird damit nicht mehr die Basisadresse \$Cn00, sondern \$Cn07 aufgerufen. Für Zeicheneingabe wird entsprechend mit \$Cn05 begonnen. Die Initialisierung der Schnittstellen-Hardware wird dann bei \$C800 fortgesetzt.

Bei Druckern ist es sinnvoll, an dieser Stelle auch gleich einen Zeichenzähler für die Zeilenlänge zu initialisieren. In „ZEILBR“ ist dies mit 80 Zeichen für normale Papierbreite vorgesehen. Soll

keine Zeilenbeschränkung erfolgen (z. B. Modems, Plotter), so ist dieses Register mit „00“ zu laden.

Gleichschritt für Bildschirm und Drucker

Soll die Universalschnittstelle einen Drucker unter Basic betreiben, so wird man sehr schnell feststellen, daß die Tabulatorfunktionen (z. B. TAB, Komma) in Print-Befehlen nicht auf das Druckbild übertragen werden und andererseits der 40-Zeichen-Bildschirm an den unpassendsten Stellen ein „unechtes“ CR (Carriage Return) einfügt.

Um diesem Mangel abzuweichen, muß die Synchronität zwischen der horizontalen Cursor-Stellung im Register „CH“ (\$24) und einem Zeichenzähler der Schnittstelle erzwungen werden. Dies erledigt die Programmschleife von INCSPC bis KORR24. Die eigentliche Ausgabe der Zeichen geschieht dann mit OUTPUT bei \$C9AA, der Einlesevorgang bei \$C841 mit INPUT, was auch den Erfordernissen bei Pascal und CP/M entspricht.

Soweit das Basic-Interface. Bei der Ausgabe bzw. Eingabe über Pascal wird aber dieses mühevoll erstellte Programm in der I/O-Page überhaupt nicht durchlaufen.

Beim Apple-Pascal ist alles ganz anders

Als seinerzeit das Pascal-Betriebssystem erstmals auf dem Apple implementiert wurde, lieferte die Firma Apple nur vier verschiedene Schnittstellen (Disk-Controller, Communication-Card, Serial-Card, Parallel-Card), und das BIOS wurde so ausgelegt, daß diese Karten direkt und ohne Umweg über die I/O-Pages \$Cn00...\$CnFF bedient wurden. Dies ist auch nicht notwendig, da nach UCSD-Standard die Formatierung und Initialisierung von einer höheren Programmebene gesteuert werden. Leider hatte man nur vergessen, dem Benutzer die Möglichkeit zu geben, einen eigenen Schnittstellentyp (Firmware) zu implementieren.

Einzig und allein die „Serial Card“ bietet hier eine Möglichkeit, da die Bedienung des Schnittstellen-Bausteins dem Programm im Erweiterungs-ROM (\$C800...\$CFFF) überlassen wurde. Hierbei werden im wesentlichen drei Einsprungadressen bedient:

\$C800 Initialisierung der Schnittstelle
\$C84D Einlesen eines ASCII-Zeichens, mit Übergabe in \$5B8 + \$Cn
\$C9AA Ausgabe eines ASCII-Zeichens im Akku bzw. \$5B8 + \$Cn

In allen Fällen stellt das BIOS die Slotnummer im Y-Register und in \$6F8 in der Form \$n0 und im X-Register in der Form \$Cn zur Verfügung.

Pascal-Version 1.1 mit „Firmware-Schnittstelle“

Erst mit der Pascal-Version 1.1 besteht nun die Möglichkeit, eine sogenannte Firmware-Karte ganz nach den Bedürfnissen des Benutzers zu programmieren, ohne irgendwelche Einschränkungen hinnehmen zu müssen. Unschön war z. B. bei der Version 1.0 der Verzicht auf das „type ahead“ für die „Console“ im Slot 3, das sehr angenehm ist, da hiermit die Kommandozeichen schneller eingegeben werden können, als sie abgearbeitet werden.

Wer aber noch mit der alten Pascal-Version arbeitet, sollte auch auf seine Kosten kommen, und deshalb wurde das Programm so ausgelegt, daß es für diese Benutzer als „falsche“ serielle Schnittstelle arbeitet, während für das neue Betriebssystem die erweiterten Möglichkeiten genutzt werden können. Dies funktioniert dann gleichermaßen unter Apple-Fortran und CP/M. Bei Speicherplatzproblemen kann man dann bei Version 1.1 auf die feste Anbindung an die Einsprungadressen verzichten.

Wie erkennt Pascal seine Schnittstellen?

Erst einmal muß das BIOS feststellen, ob überhaupt eine Karte in irgendeinem Slot sitzt, dann muß es überprüfen, um welchen Schnittstellentyp es sich handelt. Der erste Zeitpunkt, zu dem dies geschieht, ist der Boot-Vorgang (aber auch beim „Volume“-Befehl des Filers usw.). Zuerst wird der Bereich von \$C100...\$C7FF seitenweise daraufhin untersucht, ob irgendein Programmspeicher vorhanden ist, indem zweimal eine Prüfsumme gebildet wird. Eventuell vorhandene RAM-Speichereinhalte werden nicht zerstört. Weichen die beiden Prüfsummen voneinander ab, nimmt das Programm an, daß sich keine Schnittstellen-ROMs in diesem Bereich befinden.

Ist aber auf diese Art die Existenz einer Schnittstelle nachgewiesen, wird anhand einer Tabelle verglichen, um welchen Typ es sich handelt. Hierbei werden nur die Bytes in \$Cn05 und \$Cn07 herangezogen. Es gelten die Zuordnungen, die in der Tabelle aufgeführt sind. Die Unterscheidung zwischen serieller Karte und Firmware-Karte kann Pascal dadurch vornehmen, daß in \$Cn0B ein Signaturbyte abgefragt wird, das bei der Firmware-Karte \$01 sein muß, andernfalls wird die serielle Karte angenommen.

Der nächste Speicherplatz \$Cn0C soll ein weiteres Signaturbyte enthalten, das den Schnittstellentyp charakterisiert, da ja z. B. als Firmware-Karte auch eine Parallel-Karte mit einem anderen Protokoll denkbar wäre. Für ein seriell Interface sei hier \$31 empfohlen (weitere Hinweise in [2]). Der tatsächliche Wert ist aber unkritisch, da die Apple-Pascal-Vers. 1.1 dieses zweite Signaturbyte nicht auswertet. Die nachfolgenden vier Bytes sind dann allerdings von besonderer Wichtigkeit. Sie enthalten die Offsets zu den Programmanfängen:

\$Cn0D: Offset für Initialisierung
 \$Cn0E: Offset zum Eingabeprogramm
 \$Cn0F: Offset zum Ausgabeprogramm
 \$Cn10: Offset zur Statusroutine

Der jeweilige Offset besteht aus einem Byte und ergibt z. B. für (\$Cn0F) = \$MM einen Einsprungpunkt mit \$CnMM für die Schreibroutine.

Gegenüber der alten Pascal-Version 1.0 ergeben sich zwei wesentliche Vorteile:

- Für kleine Treiberprogramme reicht ein 256-Byte-Programmspeicher in \$Cn00...\$CnFF aus.
- Neben Ausgabe, Eingabe und Initialisierung steht eine weitere Funktion, nämlich die Statusabfrage zur Verfügung.

Codierung der Schnittstellentypen für das Pascal-BIOS

Schnittstellentyp	(\$Cn05)	(\$Cn07)
Karte nicht erkannt		
Disketten-Controller	\$03	\$3C
Kommunikations-Karte	\$18	\$38
Serielle Karte	\$38	\$18
Parallel-Karte	\$48	\$48
Firmware-Karte	\$38	\$18

Für alle Einsprungpunkte gilt nun folgende Vereinbarung: Um den Bereich von \$C800...\$CFFF effektiv nutzen zu können, stellt das BIOS im X-Register (\$Cn) und im Y-Register (\$n0) wieder die Slot-Nr. zur Verfügung und setzt das Enable-Flipflop. Damit sind die langwierigen Berechnungen der Slotnummern unter Pascal nicht mehr nötig. Vor Verlassen der Routinen muß allerdings das X-Register mit einem Fehlercode geladen werden, der in der Pascal-Ebene in die Variable „IORESULT“ übernommen wird. Tritt kein Fehler auf oder wird eine Fehlerprüfung nicht vorgenommen, sollte X = 0 gesetzt werden [3].

Für die Eingaberoutine gilt nun, daß das gelesene ASCII-Zeichen im Akku an das aufrufende Programm zu übergeben ist. Das Zeichen muß nicht auch noch in \$678 + \$0n abgelegt werden. Es ist aber empfehlenswert, dies zu tun, um Kompatibilität mit Version 1.0 herzustellen. Der Inhalt des Y-Registers muß nicht erhalten bleiben.

Bei der Ausgaberroutine wird das ASCII-Zeichen vom BIOS im Akku übergeben. Für die Register „X“ und „Y“ gelten die gleichen Vereinbarungen wie bei der Eingaberoutine.

Der Statusroutine hingegen wird vom BIOS im Akku ein Codewort übergeben, das für

Akku = 0 danach fragt, ob die Schnittstelle ein Zeichen ausgeben kann, für

Akku = 1 danach fragt, ob die Schnittstelle für die Eingabe ein Zeichen bereitgestellt hat.

Die Antwort auf die Statusabfrage muß im Carry-Bit des Prozessorstatus mit „Carry Clear“ beantwortet werden, wenn kein Zeichen zur Verfügung steht oder keine Ausgabe gemacht werden kann, andernfalls ist das Carry-Bit zu setzen. Wichtig ist allerdings, daß der Inhalt des Y-Registers erhalten bleibt.

Am Rande sei noch erwähnt, daß für zukünftige Pascal-Implementationen auch die Bedienung von Interrupt-Routinen bei den Schnittstellen vorgesehen ist und hierfür Offsets in ähnlicher Weise wie oben für \$Cn12 (Steueraufruf) und \$Cn13 (Statustest, „Polling“) reserviert sind. Ob die Karte interruptfähig ist, wird mit \$Cn11 kenntlich gemacht, indem dort \$00 abgelegt ist. Da diese Interruptverarbeitung für die Pascal-Ver-

sion 1.1 noch nicht zur Verfügung steht, sollte man dafür sorgen, daß an dieser Stelle keine Null steht.

Man beachte, daß die Firmware-Karte unter Version 1.0 als serielle Schnittstelle erkannt wird. Deshalb sollten die Offsets letztlich auf JMP-Befehle zeigen, die ihrerseits die Speicherplätze \$C800 (Initialisierung), \$C84D (Input) und \$C9AA (Output) anspringen, damit beim älteren Betriebssystem keine Schwierigkeiten auftreten.

Noch ein paar nützliche Hinweise

- Das Pascal-System geht davon aus, daß das höchstwertige Bit der ASCII-Zeichen, die über die „Console:“ eingegeben werden, zurückgesetzt ist (beim Apple-Basic bzw. Monitor ist hingegen das MSB gesetzt).
- Die mit der Firmware-Karte mögliche Statusroutine sollte so schnell wie möglich ablaufen, höchstens jedoch 100 ms dauern.
- Wird die Firmware-Karte in Slot 1 gesteckt, so wird sie automatisch als „Printer:“ nur für Ausgabe betrieben. Eine Firmware-Karte in Slot 2 wird als „Remin:“ und „Remout:“ bidirektional bedient, und in Slot 3 nimmt die Karte die Funktionen der „Console:“ bzw. des „System:“ wahr.
- Das Pascal-Betriebssystem erzeugt automatisch nach jedem „CR“ ein „LF“, wenn die Schnittstelle in den genannten zeichenorientierten Slots betrieben wird. Demnach muß eventuell die Auto-LF-Option eines Druckers abgeschaltet werden oder das LF schon in der Schnittstelle unterdrückt werden, was aber nicht immer zum gleichen Ergebnis führt.
- Für Betreiber des Basis 108 sei angemerkt, daß bei einigen Rechnern Schreibfehler im RAM auftraten, die sich mit einer ODER-Verknüpfung (R/W mit verzögertem $\Phi 1$ für das WR-Signal des Speichers) beheben lassen. Der Apple-II benötigt dieses Gatter nicht.

Literatur

- [1] APPLE II Reference Manual. Apple Inc.
- [2] Haynes, Barry: ATTACH-BIOS document for APPLE II PASCAL Version 1.1.
- [3] APPLE Pascal, Language Reference Manual. Apple Inc.

Herwig Feichtinger

Kommunikation mit dem Apple-II

Um den Apple-II (oder einen seiner kompatiblen Brüder) in ein Terminal umzuwandeln, so daß man z. B. mit einer Datenbank oder dem Franzis-Teledaten-Service TEDAS per Telefonmodem Kontakt aufnehmen kann, ist zweierlei nötig – eine Duplex-V.24-Schnittstelle und geeignete Terminal-Software. Der folgende Beitrag stellt für beides eine preiswerte Lösung vor.

Die für den Apple-II für weniger als 200 DM erhältliche „Serial Interface Card“ ist für den Betrieb als Duplex-V.24-Schnittstelle aus Hardware-Gründen leider nicht geeignet, da Zeichen nicht unabhängig voneinander gleichzeitig empfangen und gesendet werden können.

Einfache Hardware...

Es wurde hier daher eine andere Interface-Karte gewählt, die preislich in der gleichen Größenordnung liegt, aber wesentlich universeller verwendbar ist, nämlich die 6532-Karte von Neucrom (Hangweg 4, 8893 Hilgertshausen-Tandern). Die DIL-Schalter auf ihr sind in Stellung „EMUF“ zu bringen, so daß der

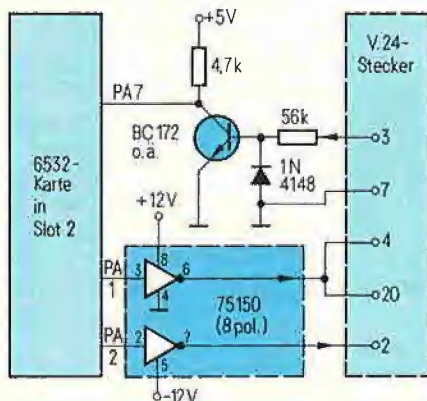


Bild 1. Interface-Schaltung für eine V.24-Schnittstelle. Ein Programmtest ist durch Verbinden der Steckerpins 2 und 3 möglich

volle 6532-Adressenbereich aktiviert ist und sich der Timer-Interrupt softwaremäßig ein- und ausschalten läßt. Benutzt werden dann folgende Adressen (die Karte steckt in Slot 2 des Apple): C280 = PA-Datenregister; C281 = PA-Datenrichtungsregister; C29E = Timer 64 µs mit IRQ; C296 = Timer lesen/IRQ ausschalten.

Um die nötigen V.24-Verbindungen hergestellt zu werden, ist die kleine Schaltung in *Bild 1* erforderlich. An sie kann unmittelbar ein Modem angeschlossen werden. Über die Stifte 4 und 20 des V.24-Steckers wird ein galvanisch gekoppeltes Modem eingeschaltet; bei einem Akustikkoppler könnte man auf diese Verbindungen auch verzichten. Der Datentransfer erfolgt über die

V.24-Stifte 2 (zum Modem) und 3 (vom Modem) mit den Pegeln -12 V für 1 und $+12\text{ V}$ für 0. Die Pinbelegung des V.24-Treibers 75150 ist für die 8polige Version gezeichnet.

...komfortable Software

Da die 6532-Karte nur I/O-Ports und einen Timer, aber kein UART oder Schieberegister enthält, wurde die serielle Ein- und Ausgabe rein softwaremäßig realisiert. Der Empfang von Zeichen (und das ist der eigentliche Trick!) erfolgt sozusagen im „Hintergrund“ über eine periodisch angesprungene Interrupt-Routine, die nahezu unverändert einem 6502-Buch entnommen wurde [1]. Sie legt die vom Modem kommenden Zeichen in einem Pufferbereich ab (hex 0F00...0FFF), die 255 Zeichen umfaßt. Das Hauptprogramm holt die Zeichen aus diesem Puffer wieder heraus und stellt sie auf dem Apple-II-Bildschirm dar. Diese Methode ist aus zwei Gründen nötig: Zum einen braucht der Apple z. B. für einen Sprung zur nächsten Zeile zu lange; der Zwischenraum zwischen zwei empfangenen Zeichen würde dafür nicht ausreichen. Zum anderen würden ohne Puffer beim Aufruf von Steuerfunktionen, auf die wir noch zu sprechen kommen, Zeichen verlorengehen.

Die Routine zum seriellen Senden von ASCII-Zeichen ist [2] entnommen und berücksichtigt zusätzlich, daß manche Datenbank-Systeme, auch TEDAS, nach einem Return-Zeichen eine Pause von einigen 100 ms benötigen, um eine Zeile zu verarbeiten.

Die Bedienung

Der Start des Terminal-Programms erfolgt von Basic aus mit CALL 576 oder

Escape-Steuersequenzen des Terminalprogramms

ESC 0	Modem ausschalten
ESC 1	Modem einschalten
ESC H	Zurück zu Basic
ESC S	Empfangenen Text ab hex 1000 abspeichern (Download)
ESC ESC	Abspeichern beenden
ESC U	Ab hex 1000 gespeicherten Text senden (Upload)
ESC D	Ab hex 1000 gespeicherten Text anzeigen (Display)

Hinweis: Für den der Escape-Taste folgenden Buchstaben ist Groß- und Kleinschreibung zulässig (vgl. Seite 75).

vom Monitor aus mit 240G. Nur dann, wenn von Basic aus gestartet wurde (!), kommt man vom Terminal-Programm mit ESC H in Basic zurück. Wenn das Terminal-Programm einmal initialisiert ist, kann man dann z. B. in Basic zurück-

gehen (ESC H) und mit CALL 579 später einen Warmstart durchführen, denn die Interrupt-Routine läuft währenddessen ja weiter und legt empfangene Zeichen im Puffer ab. Den Interrupt sollte man allerdings vor Disketten-Operationen

ausschalten, um Aufzeichnungs- und Lesefehler zu vermeiden; das kann z. B. mit dem Befehl A=PEEK(-15722) geschehen, da damit die Timer-Adresse C296 gelesen und der Interrupt gesperrt wird. Auch dann ist ein Einsprung über

0240-	20 BF 03	JSR	\$03BF	02C3-	C9 DE	CMP	£\$DE	034D-	DO 11	BNE	\$0360
0243-	4C FF 02	JMP	\$02FF	02C5-	DO 02	BNE	\$02C9	034F-	2C 80 C2	BIT	\$C280
0246-	A0 00	LDY	£\$00	02C7-	A9 CE	LDA	£\$CE	0352-	10 04	BPL	\$0358
0248-	B1 3C	LDA	(\$3C),Y	02C9-	C9 C0	CMP	£\$C0	0354-	A9 10	LDA	£\$10
024A-	C9 8D	CMP	£\$8D	02CB-	DO 02	BNE	\$02CF	0356-	DO 2A	BNE	\$0382
024C-	DO 05	BNE	\$0253	02CD-	A9 DO	LDA	£\$DO	0358-	A9 09	LDA	£\$09
024E-	A9 FF	LDA	£\$FF	02CF-	C9 DD	CMP	£\$DD	035A-	85 FC	STA	\$FC
0250-	20 A8 FC	JSR	\$FCA8	02D1-	DO 02	BNE	\$02D5	035C-	A9 46	LDA	£\$46
0253-	4C C2 FC	JMP	\$FCC2	02D3-	A9 CD	LDA	£\$CD	035E-	DO 22	BNE	\$0382
0256-	A2 FF	LDX	£\$FF	02D5-	C9 9B	CMP	£\$9B	0360-	C6 FC	DEC	\$FC
0258-	86 3D	STX	\$3D	02D7-	DO 42	BNE	\$031B	0362-	FO 09	BEQ	\$036D
025A-	E8	INX		02D9-	20 8A 03	JSR	\$038A	0364-	AD 80 C2	LDA	\$C280
025B-	86 FB	STX	\$FB	02DC-	C9 10	CMP	£\$10	0367-	0A	ASL	
025D-	60	RTS		02DE-	DO 04	BNE	\$02E4	0368-	66 FD	ROR	\$FD
025E-	24 FB	BIT	\$FB	02EO-	A9 06	LDA	£\$06	036A-	4C 80 03	JMP	\$0380
0260-	30 05	BMI	\$0267	02E2-	DO 06	BNE	\$02EA	036D-	A5 FD	LDA	\$FD
0262-	48	PHA		02E4-	C9 11	CMP	£\$11	036F-	09 80	ORA	£\$80
0263-	20 78 FB	JSR	\$FB78	02E6-	DO B6	BNE	\$029E	0371-	C9 8A	CMP	£\$8A
0266-	68	PLA		02E8-	A9 04	LDA	£\$04	0373-	FO 0B	BEQ	\$0380
0267-	60	RTS		02EA-	8D 80 C2	STA	\$C280	0375-	84 FA	STY	\$FA
0268-	A9 00	LDA	£\$00	02ED-	4C 10 03	JMP	\$0310	0377-	E6 FF	INC	\$FF
026A-	85 3C	STA	\$3C	02F0-	24 3D	BIT	\$3D	0379-	A4 FF	LDY	\$FF
026C-	A9 10	LDA	£\$10	02F2-	30 37	BMI	\$032B	037B-	99 00 OF	STA	\$0F00,Y
026E-	85 3D	STA	\$3D	02F4-	A0 00	LDY	£\$00	037E-	A4 FA	LDY	\$FA
0270-	60	RTS		02F6-	91 3C	STA	(\$3C),Y	0380-	A9 33	LDA	£\$33
0271-	EA	NOP		02F8-	98	TYA		0382-	8D 9E C2	STA	\$C29E
0272-	EA	NOP		02F9-	C8	INY		0385-	A5 45	LDA	\$45
0273-	48	PHA		02FA-	91 3C	STA	(\$3C),Y	0387-	40	RTI	
0274-	20 9C FC	JSR	\$FC9C	02FC-	4C C2 FC	JMP	\$FCC2	0388-	EA	NOP	
0277-	68	PLA		02FF-	58	CLI		0389-	EA	NOP	
0278-	60	RTS		0300-	20 56 02	JSR	\$0256	038A-	20 0C FD	JSR	\$FDOC
0279-	20 68 02	JSR	\$0268	0303-	A9 4B	LDA	£\$4B	038D-	29 1F	AND	£\$1F
027C-	20 58 FC	JSR	\$FC58	0305-	8D FE 03	STA	\$03FE	038F-	60	RTS	
027F-	A0 00	LDY	£\$00	0308-	A9 03	LDA	£\$03	0390-	29 7F	AND	£\$7F
0281-	B1 3C	LDA	(\$3C),Y	030A-	8D FF 03	STA	\$03FF	0392-	20 AF 03	JSR	\$03AF
0283-	FO 30	BEQ	\$02B5	030D-	8D 9E C2	STA	\$C29E	0395-	A2 0B	LDX	£\$0B
0285-	20 5E 02	JSR	\$025E	0310-	AD 00 C0	LDA	\$C000	0397-	38	SEC	
0288-	24 FB	BIT	\$FB	0313-	10 09	BPL	\$031E	0398-	6A	ROR	
028A-	10 0C	BPL	\$0298	0315-	2C 10 C0	BIT	\$C010	0399-	90 05	BCC	\$03A0
028C-	20 90 03	JSR	\$0390	0318-	4C C3 02	JMP	\$02C3	039B-	20 A7 03	JSR	\$03A7
028F-	EA	NOP		031B-	20 90 03	JSR	\$0390	039E-	30 03	BMI	\$03A3
0290-	EA	NOP		031E-	20 2B 03	JSR	\$032B	03A0-	20 AF 03	JSR	\$03AF
0291-	EA	NOP		0321-	4C 10 03	JMP	\$0310	03A3-	CA	DEX	
0292-	EA	NOP		0324-	EA	NOP		03A4-	DO F1	BNE	\$0397
0293-	EA	NOP		0325-	EA	NOP		03A6-	EA	NOP	
0294-	EA	NOP		0326-	EA	NOP		03A7-	48	PHA	
0295-	20 2B 03	JSR	\$032B	0327-	EA	NOP		03A8-	AD 80 C2	LDA	\$C280
0298-	20 46 02	JSR	\$0246	0328-	EA	NOP		03AB-	09 04	ORA	£\$04
029B-	4C 7F 02	JMP	\$027F	0329-	EA	NOP		03AD-	DO 06	BNE	\$03B5
029E-	C9 15	CMP	£\$15	032A-	EA	NOP		03AF-	48	PHA	
02A0-	DO 06	BNE	\$02A8	032B-	A4 FE	LDY	\$FE	03B0-	AD 80 C2	LDA	\$C280
02A2-	A9 80	LDA	£\$80	032D-	C4 FF	CPY	\$FF	03B3-	29 FB	AND	£\$FB
02A4-	85 FB	STA	\$FB	032F-	DO 01	BNE	\$0332	03B5-	8D 80 C2	STA	\$C280
02A6-	30 D1	BMI	\$0279	0331-	60	RTS		03B8-	A9 21	LDA	£\$21
02A8-	C9 13	CMP	£\$13	0332-	E6 FE	INC	\$FE	03BA-	20 A8 FC	JSR	\$FCA8
02AA-	DO 05	BNE	\$02B1	0334-	A4 FE	LDY	\$FE	03BD-	68	PLA	
02AC-	20 68 02	JSR	\$0268	0336-	B9 00 OF	LDA	\$0F00,Y	03BE-	60	RTS	
02AF-	DO 3C	BNE	\$02ED	0339-	48	PHA		03BF-	A2 06	LDX	£\$06
02B1-	C9 1B	CMP	£\$1B	033A-	20 FD FB	JSR	\$FBFD	03C1-	8E 81 C2	STX	\$C281
02B3-	DO 06	BNE	\$02BB	033D-	68	PLA		03C4-	8E 80 C2	STX	\$C280
02B5-	20 56 02	JSR	\$0256	033E-	C9 88	CMP	£\$88	03C7-	86 FE	STX	\$FE
02B8-	4C 10 03	JMP	\$0310	0340-	DO 03	BNE	\$0345	03C9-	86 FF	STX	\$FF
02BB-	C9 04	CMP	£\$04	0342-	20 73 02	JSR	\$0273	03CB-	60	RTS	
02BD-	FO BA	BEQ	\$0279	0345-	24 FB	BIT	\$FB	03CC-	EA	NOP	
02BF-	DO 70	BNE	\$0331	0347-	10 A7	BPL	\$02FO	03CD-	EA	NOP	
02C1-	EA	NOP		0349-	30 E0	BMI	\$032B				
02C2-	EA	NOP		034B-	A5 FC	LDA	\$FC				

Bild 2. Listing des Apple-II-Programms zur Emulation eines Duplex-Terminals. Es ist auch auf Diskette vom Franzis-Software-Service zu beziehen

die Warmstartadresse aber möglich, da der Interrupt auf jeden Fall neu gestartet wird (lediglich die Pufferzeiger und die Modem-Steuerleitungen werden beim Warmstart nicht initialisiert).

Nach einem Kaltstart mit CALL 576 ist das Modem zunächst ausgeschaltet. ESC 1 schaltet es ein (bei einem Akustikkoppler ist das allerdings meist überflüssig), mit ESC 0 kann man es nach Beenden der Verbindung wieder von der Fernsprechleitung trennen. Weitere Escape-Sequenzen sind in der Tabelle aufgeführt.

Sobald man auf die Escape-Taste drückt, erscheint ein blinkender Cursor als Aufforderung, eine weitere Taste als Steuerfunktion zu drücken. Nach ESC S wird der empfangene Text im Speicherbereich hex 1000...7FFF mitgeschrieben und läßt sich von Basic aus per BSAVE abspeichern. Die Endadresse des Textes läßt sich ermitteln, indem man das erste Auftreten eines Null-Bytes sucht; es dient als Textende-Markierung. Alle ASCII-Zeichen werden mit gesetztem Bit 7 abgespeichert, ein „A“ ist also hex C1.

Bild 2 zeigt auch ein Basic-Rahmenprogramm, das das Suchen der Text-Endadresse, das Abspeichern auf Diskette, das Wiederladen des Textes und den Start des Terminal-Programms benutzerfreundlich übernimmt. Auch könnte man sich in Basic ein einfaches Editor-Programm erstellen, um Texte in den Speicher schreiben und dann mit der Upload-Funktion absenden zu können. Am Anfang dieser Basic-Programme sollte mit HIMEM:4095 verhindert werden, daß Strings den Textspeicher-Bereich überschreiben können.

Betriebsparameter

Das Terminalprogramm besitzt fest eingestellte Betriebsparameter, die sich an gängigen Datenbank-Systemen bzw. dem Franzis-Teledaten-Service TEDAS (Tel. 0 89/59 64 22) orientieren. Die Übertragung (300 Bd) erfolgt vollduplex mit je acht Datenbits pro Zeichen und ohne Parität. Empfangsseitig ist mindestens ein Stopbit nötig, sendeseitig werden jedoch drei erzeugt, um der Gegenstation genug Zeit zur Verarbeitung zu

lassen. Ferner wird nach jedem gesendeten Return-Zeichen eine Pause von rund 200 ms eingefügt, was insbesondere bei der Upload-Funktion wichtig ist. Nach Return wird kein Line-Feed gesendet, und empfangene Line-Feed-Zeichen werden unterdrückt, da beim Apple-II auch Return schon einen Zeilenvorschub bewirkt.

Ein Erhöhen der Baudrate auf 1200 Bd ist leider nicht ohne weiteres möglich, da dann der prozentuale Zeitbedarf der Interrupt-Empfangsroutine zu einer zu großen Änderung der V.24-Sendezeitkonstanten (Sender-Baudrate) führen würde, die mit einer Verzögerungsschleife des Apple-Monitorprogramms realisiert ist.

Literatur

- [1] Feichtinger, H.: Programmierpraxis mit dem 6502. Franzis-Verlag, München 1983.
- [2] Feichtinger, H.: Mit Computern steuern. Franzis-Verlag, München 1983.

Routinen des Terminal-Programms

```

240 Kaltstartadresse
243 Warmstartadresse
246 Verzög., wenn Zeichen=CR;Pointer+1
256 Upload und Download ausschalten
25E Zeichen anz., falls nicht Upload
268 Textzeiger auf $1000 initialisieren
273 Rest der Zeile löschen, Akku retten
279 Routine für Display und Upload
29E Escape-Sequenz-Decodierung
2C3 Umcodierung f. Shift-N, Shift-P usw.
2D9 Escape-Sequenzen: Modem aus/ein
2F0 Zeichen abspeichern, falls Download
2FF Interrupt-Initialisierung
310 Tastatur abfragen (Hauptschleife)
32B Zeichen aus Empfangspuffer ausgeben
34B Interrupt-Routine f. Empfang
38A Zeichen f. Escape-Sequenz einlesen
390 Zeichen an V.24 ausgeben
3BF 6532-Port A und Pufferzeiger init.
```

Verwendete Zero-Page-Adressen

```

3C-3D Textzeiger und Flag f. Download
FA Zwischenspeicher für Y-Register
FB Flag für Upload
FC Bitzähler für Interruptroutine
FD Empfangenes Zeichen
FE Pufferzeiger f. Hauptprogramm
FF Pufferzeiger f. Interrupt-Routine
```

Basic-Hilfsprogramm für die Terminal-Software

```

10REM NEUCOM-KARTE IN STELLG.EMUF,SL.2
70HIMEM:3840:D$=CHR$(4)
75IFPEEK(576)=32ANDPEEK(577)=191ANDPEEK(971)
)=96THEN100
80PRINTCHR$(4)"BLOADMODEM":REM Maschinenprogramm
```

```

100HOME:PRINT"TEDAS-KOMMUNIKATIONSPROGRAMM
105PRINT:PRINT"(C) Herwig Feichtinger 1983
110VTAB$="PRINT"Funktionen im Terminalmodus":PRINT
120PRINT"ESC-1 Modem anschalten
130PRINT"ESC-0 Modem abtrennen
135PRINT"ESC-S Empfangenen Text speichern
140PRINT"2xESC Abspeichern beenden
150PRINT"ESC-U Gespeicherten Text senden
160PRINT"ESC-D Gespeicherten Text anzeigen
190PRINT"ESC-H Hierher zurueck":PRINT
200PRINT:PRINT"Return -> Terminalmodus
202PRINT:PRINT"K Dieses Programm kopieren
203PRINT"L Text von Disk laden
204PRINT"S Text auf Disk speichern":PRINT
210GETA$:IFASC(A$)=13THEN235
211A=PEEK(49814):REM .IRQ AUS
213IFA$="K"THEN250
215IFA$="S"THEN400
217IFA$="L"THEN420
220PRINT"Applesoft";:END
235HOME:A=PEEK(49792):REM PA1=?
237IFA=125ORA=253THENCALL579:RUN
240PRINT"TEDAS: Tel. 089-596422":CALL576:RUN
250HOME:PRINT"TEDAS und MODEM werden kopiert."
252PRINT"Initialisierte Disk einlegen"
255PRINT"und RETURN druecken: ";
260GETA$:IFASC(A$)<>13THENRUN
270PRINT:PRINT"Einen Moment..."
280PRINTD$;"SAVETEDAS"
290PRINTD$;"BSAVEMODEM,A$240,L$190"
300RUN
400PRINT"Moment...":I=4096
401IFPEEK(I)>0THENI=I+1:GOTO401
402IFI<4130THENRUN
405HOME:INPUT"Text abspeichern; Name: ";A$
410PRINTD$;"BSAVE";A$;"",A4096,L";I=4095
415RUN
420HOME:INPUT"Text laden; Name: ";A$
425PRINTD$;"BLOAD";A$
430RUN
```

Herwig Feichtinger

Bytefolgen schnell gefunden

Wer Maschinenprogramme durchforstet, von denen er kein Assembler-Quellenlisting besitzt und die er vielleicht an einen anderen Computer anpassen will, wird es schnell mühsam finden, jene Stellen zu suchen, wo z. B. Systemaufrufe vorkommen oder bestimmte Speicherzellen angesprochen werden. Das folgende Maschinenprogramm löst das Problem zumindest beim Apple-II.

Das in Bild 1 als Assembler-Protokoll abgedruckte Apple-II-Programm verwendet folgende systemspezifische Adressen: GETLN ist ein Unterprogramm im Apple-ROM, das eine Zeile von der Tastatur holt (bis zum Druck auf Return) und die ASCII-Zeichen mit Bit 7 = 1 ab Adresse 0200 ablegt (inklusive

Return). Die gedrückten Tasten werden auf dem Bildschirm angezeigt, und man kann mit dem Cursor korrigieren. INBUF ist der gerade erwähnte Puffer für GETLN. OUT ist eine ROM-Routine, die ein ASCII-Zeichen im Akku (Bit 7 = 1) auf den Bildschirm ausgibt. PRAX stellt Akku und X-Register hexadezimal auf

dem Bildschirm dar, also als vierstellige Hexzahl. MON ist die Einsprungsadresse in den Apple-II-Monitor, und CTRL-Y ist ein Vektor im RAM, über den bei Druck auf CTRL-Y zu einem Anwenderprogramm gesprungen werden kann, hier zur Routine SRCH.

Alle übrigen Labels werden nur vom Suchprogramm selbst benutzt. Es setzt beim Start zunächst den CTRL-Y-Vektor, so daß man fortan durch Druck auf CTRL-Y die Suchroutine aufrufen kann. Diese wiederum fragt nacheinander hexadezimal Anfangs- und Endadresse des zu durchsuchenden Bereichs und die gewünschte Bytefolge ab. Die einzelnen Bytes dürfen, müssen aber nicht durch Leerräume getrennt werden. Die Adressen müssen stets vierstellig eingegeben werden. Nach dem ersten Aufruf der Routine darf man aber statt einer Adressenangabe auch einfach nur auf Return drücken; dann wird der zuletzt angegebene Adressenbereich weiterverwendet. Das Programm gibt dann alle die Adressen aus, ab denen die Bytefolge gefunden wurde, und zwar in sechs Bildschirmspalten.

Die Eingabe des Programms kann so erfolgen:

0800	1	;BYTE-SUCHER (C)MC/FE	0844	AD8502	42	LDA BYTES
0800	2	COLM EPZ \$FF	0847	8D8302	43	STA END+1
0800	3	PNT EPZ \$FD	084A		44	;BYTEFOLGE EINLESEN
0800	4	BEG EQU \$280	084A	A218	45	SRCH0 LDX #TEXT2-TEXT
0800	5	END EQU \$282	084C	20F308	46	JSR PRINT
0800	6	LEN EQU \$284	084F	20AC08	47	JSR GET
0800	7	BYTES EQU \$285	0852	A900	48	LDA #0
0800	8	CTRL-Y EQU \$3F9	0854	85FF	49	STA COLM
0800	9	INBUF EQU \$200	0856	AD8002	50	LDA BEG
0800	10	OUT EQU \$FDED	0859	85FD	51	STA PNT
0800	11	PRAX EQU \$F941	085B	AD8102	52	LDA BEG+1
0800	12	GETLN EQU \$FD6F	085E	85FE	53	STA PNT+1
0800	13	MON EQU \$FF69	0860		54	;SPEICHER DURCHSUCHEN
0800	14	ORG \$800	0860	AC8402	55	SRCH1 LDY LEN
0800	15	;CTRL-Y AKTIVIEREN	0863	88	56	SRCH2 DEY
0800	A912	LDA #SRCH	0864	3021	57	BMI EQUAL
0802	8DF903	STA CTRL-Y	0866	B1FD	58	LDA (PNT),Y
0805	A908	LDA /SRCH	0868	D98502	59	CMP BYTES,Y
0807	8DFA03	STA CTRL-Y+1	086B	F0F6	60	BEQ SRCH2
080A	A22E	LDX #TEXT4-TEXT	086D	E6FD	61	SRCH3 INC PNT
080C	20F308	JSR PRINT	086F	D002	62	BNE SRCH4
080F	4C69FF	JMP MON	0871	E6FE	63	INC PNT+1
0812		;BEREICH EINLESEN	0873	A5FD	64	SRCH4 LDA PNT
0812	A200	SRCH LDX #0	0875	CD8202	65	CMP END
0814	20F308	JSR PRINT	0878	90E6	66	BCC SRCH1
0817	AD8102	LDA BEG+1	087A	A5FE	67	LDA PNT+1
081A	AE8002	LDX BEG	087C	CD8302	68	CMP END+1
081D	20AC08	JSR GET	087F	90DF	69	BCC SRCH1
0820	F028	BEQ SRCH0	0881	4C69FF	70	JMP MON
0822	AD8502	LDA BYTES	0884	4C6008	71	JMP SRCH1
0825	8D8102	STA BEG+1	0887		72	;FOLGE GEFUNDEN!
0828	AD8602	LDA BYTES+1	0887	A5FE	73	EQUAL LDA PNT+1
082B	8D8002	STA BEG	0889	A6FD	74	LDX PNT
082E	A20D	LDX #TEXT1-TEXT	088B	2041F9	75	JSR PRAX
0830	20F308	JSR PRINT	088E	A9A0	76	LDA #\$A0
0833	AD8302	LDA END+1	0890	20EDFD	77	JSR OUT
0836	AE8202	LDX END	0893		78	;6 SPALTEN AUSGEBEN
0839	20AC08	JSR GET	0893	E6FF	79	INC COLM
083C	F00C	BEQ SRCH0	0895	A5FF	80	LDA COLM
083E	AD8602	LDA BYTES+1	0897	C906	81	CMP #6
0841	8D8202	STA END	0899	90D2	82	BCC SRCH3

Bild 1.
Assemblerlisting
des Bytefolgen-
Suchprogramms
für den Apple-II

Das Apple-Sonderheft

```

089B A98D 83 LDA #$8D
089D 20EDFD 84 JSR OUT
08A0 A900 85 LDA #0
08A2 85FF 86 STA COLM
08A4 4C6D08 87 JMP SRCH3
08A7 88 ;ADR./BYTES EINLESEN
08A7 A223 89 ERR LDX #TEXT3-TEXT
08A9 20F308 90 JSR PRINT
08AC 206FFD 91 GET JSR GETLN
08AF A200 92 LDX #0
08B1 A000 93 LDY #0
08B3 BD0002 94 GET1 LDA INBUF,X
08B6 E8 95 INX
08B7 C9A0 96 CMP #$A0
08B9 F0F8 97 BEQ GET1
08BB 20DC08 98 JSR PACK
08BE B01A 99 BCS GET2
08C0 0A 100 ASL
08C1 0A 101 ASL
08C2 0A 102 ASL
08C3 0A 103 ASL
08C4 85FF 104 STA COLM
08C6 BD0002 105 LDA INBUF,X
08C9 20DC08 106 JSR PACK
08CC B0D9 107 BCS ERR
08CE 05FF 108 ORA COLM
08D0 998502 109 STA BYTES,Y
08D3 E8 110 INX
08D4 C8 111 INY
08D5 8C8402 112 STY LEN
08D8 D0D9 113 BNE GET1
08DA 98 114 GET2 TYA
08DB 60 115 RTS
08DC 116 ;ASCII ZU NIBBLE
08DC C9B0 117 PACK CMP #$B0
08DE 9011 118 BCC NOHEX
08E0 C9C7 119 CMP #$C7
08E2 B00D 120 BCS NOHEX
08E4 C9BA 121 CMP #$BA
08E6 9006 122 BCC PACK1
08E8 C9C0 123 CMP #$C0
08EA 9005 124 BCC NOHEX
08EC 6908 125 ADC #8

```

```

08EE 290F 126 PACK1 AND #$F
08F0 60 127 RTS
08F1 38 128 NOHEX SEC
08F2 60 129 RTS
08F3 130 ;TEXT AUSGEBEN
08F3 BDFE08 131 PRINT LDA TEXT,X
08F6 F0F9 132 BEQ NOHEX
08F8 20EDFD 133 JSR OUT
08FB E8 134 INX
08FC D0F5 135 BNE PRINT
08FE 8D8D 136 TEXT HEX 8D8D
0900 C1C4D2 137 ASC "ADR. VON:"
0903 AEA0A0
0906 D6CFCE
0909 BA
090A 00 138 HEX 00
090B A0A0A0 139 TEXT1 ASC " BIS:"
090E A0A0A0
0911 C2C9D3
0914 BA
0915 00 140 HEX 00
0916 C2D9D4 141 TEXT2 ASC "BYTEFOLGE:"
0919 C5C6CF
091C CCC7C5
091F BA
0920 00 142 HEX 00
0921 C6C5C8 143 TEXT3 ASC "FEHLER! :"
0924 CCC5D2
0927 A1A0A0
092A BA
092B 00 144 HEX 00
092C 8D 145 TEXT4 HEX 8D
092D C3D4D2 146 ASC "CTRL-Y IST "
0930 CCADD9
0933 A0C9D3
0936 D4A0
0938 C1CBD4 147 ASC "AKTIVIERT."
093B C9D6C9
093E C5D2D4
0941 AE
0942 8D00 148 HEX 8D00
149 END

```

1. Mit CALL-151 aus Basic zum Monitor gehen.
2. Ab Adresse 0800 die Hex-Bytes in Gruppen von je etwa 16 eintippen:
0800:A9 12 8D...69 FF
0812:A2 00 usw.
3. Programm auf Diskette abspeichern:
BSAVE SRCH,A\$800,L\$144

Nach dem Start mit 800G oder BRUN SRCH meldet es sich nun mit „CTRL-Y ist aktiviert“, so daß die Suchroutine mit CTRL-Y aufgerufen werden kann. Da es zuweilen vorkommen kann, daß jenes Programm, was durchsucht werden soll, just ebenfalls bei 0800 beginnt, zeigt Bild 2 eine Programmversion als

>CALL-151

*4000G

CTRL-Y IST AKTIVIERT.

*

ADR. VON:4000

BIS:4144

BYTEFOLGE:20 ED FD

4090 409D 40F8

Bild 3. Beispiel für das Suchen nach dem Befehl JSR FDED, der der Bytefolge 20 ED FD entspricht

```

4000- A9 12 8D F9 03 A9 40 8D FA 03 A2 2E 20 F3 40 4C +=0726
4010- 69 FF A2 00 20 F3 40 AD 81 02 AE 80 02 20 AC 40 +=06C9
4020- F0 28 AD 85 02 8D 81 02 AD 86 02 8D 80 02 A2 0D +=064F
4030- 20 F3 40 AD 83 02 AE 82 02 20 AC 40 F0 0C AD 86 +=06F2
4040- 02 8D 82 02 AD 85 02 8D 83 02 A2 18 20 F3 40 20 +=0586
4050- AC 40 A9 00 85 FF AD 80 02 85 FD AD 81 02 85 FE +=087D
4060- AC 84 02 88 30 21 B1 FD D9 85 02 F0 F6 E6 FD D0 +=09B2
4070- 02 E6 FE A5 FD CD 82 02 90 E6 A5 FE CD 83 02 90 +=09D4
4080- DF 4C 69 FF 4C 60 40 A5 FE A6 FD 20 41 F9 A9 A0 +=0968
4090- 20 ED FD E6 FF A5 FF C9 06 90 D2 A9 8D 20 ED FD +=0B04
40A0- A9 00 85 FF 4C 6D 40 A2 23 20 F3 40 20 6F FD A2 +=076C
40B0- 00 A0 00 BD 00 02 E8 C9 A0 F0 F8 20 DC 40 B0 1A +=079E
40C0- 0A 0A 0A 0A 85 FF BD 00 02 20 DC 40 B0 D9 05 FF +=0634
40D0- 99 85 02 E8 C8 8C 84 02 D0 D9 98 60 C9 B0 90 11 +=089D
40E0- C9 C7 B0 0D C9 BA 90 06 C9 C0 90 05 69 08 29 0F +=072B
40F0- 60 38 60 BD FE 40 F0 F9 20 ED FD E8 D0 F5 8D 8D +=0AAD
4100- C1 C4 D2 AE A0 A0 D6 CF CE BA 00 A0 A0 A0 A0 +=0A92
4110- A0 C2 C9 D3 BA 00 C2 D9 D4 C5 C6 CF CC C7 C5 BA +=0B93
4120- 00 C6 C5 C8 CC C5 D2 A1 A0 A0 BA 00 8D C3 D4 D2 +=0A47
4130- CC AD D9 A0 C9 D3 D4 A0 C1 CB D4 C9 D6 C9 C5 D2 +=0C61
4140- D4 AE 8D 00 FF FF +=040D

```

Bild 2. Objektcode-Version mit der Startadresse hex 4000

Hexdump (mit Prüfsummen, vgl. mc 6/1984, Seite 64), die bei 4000 beginnt. Das Abspeichern erfolgt hier also mit BSAVE SRCH, A\$4000,L\$144.

Ein kleines Beispiel für die Anwendung zeigt Bild 3: Hier hat sich das Programm in Bild 2 selbst nach einem Aufruf der OUT-Routine (JSR FDED, Bytefolge 20 ED FD) durchsucht.

Das Programm befindet sich in beiden Adressenbereichs-Versionen sowie als Quellencode für den Assembler „Lisa“ von Lazer Systems bzw. kompatible Assembler auf der Apple-Sameldiskette 7 unseres Software-Service.

Heribert B. Bieling

Apple-Disk-Editor

Ob Sie nun ein gelöscht File wieder zum Leben erwecken, eine abgestürzte Diskette reparieren oder einfach nur das Liebesleben der Bytes auf der Diskette beobachten wollen, in all diesen Fällen leistet ein Disk-Editor wertvolle Hilfe.

Das vorliegende Programm wurde auf einem Apple-II-Plus entwickelt und arbeitet mit allen Laufwerken, die unter DOS 3.3 betrieben werden können. Mit Hilfe des Disk-Editors können Sie alle auf einer Diskette vorhandenen Sektoren lesen, modifizieren und wieder auf die Diskette zurückschreiben. Dabei ist es gleichgültig, ob es sich um eine DOS-, UCSD- oder CP/M-Diskette handelt.

Der Disk-Editor fragt nach dem Start zunächst nach Slot, Drive, Track und Sektor. Dabei werden nur sinnvolle Werte

zugelassen. Der Wert für den Track kann zwischen 0 und 39 liegen, um auch „aufgebohrte“ Disketten bearbeiten zu können. Der angegebene Sektor wird in den Bereich ab \$6500 eingelesen; auf dem Schirm erscheinen die ersten 128 Byte dieses Sektors in ASCII- und in Hexdarstellung. Mit den Pfeiltasten kann dann zwischen den Halbsektoren umgeblättert werden, wobei beim Überschreiten von Sektorengrenzen der nächste Sektor automatisch eingelesen wird. Auf diese Weise kann man die ganze Diskette beliebig vor- und rückwärts „scannen“. Bei

Bedarf können Sie sich auch den Bildschirminhalt mit COPY auf den Drucker ausgeben.

Die Information im angezeigten Halbsektor läßt sich beliebig überschreiben, wobei je nach Bedarf eine ASCII- oder eine Hexadezimal-Eingabe verwendet wird.

Hex- oder ASCII-Darstellung

Im Hex-Mode wird der Cursor mit der Raute I, J, K, M positioniert; Hex-Eingaben werden selbsttätig erkannt und übernommen.

Im ASCII-Mode kann mittels der ESC-Taste zwischen den Eingabearten Cursor, Normal, Klein, Invers, Flash und Control umgeschaltet werden. Der Cursor wird hierbei mit I, J, K, M über das zu überschreibende Zeichen gesetzt, dann wird mit ESC auf die entsprechende Zeichenart umgeschaltet und das gewünschte Zeichen eingegeben. Auf diese Weise lassen sich ganze Texte auf der Diskette ändern, z. B. um in einem Programm die englischen Kommentare zu übersetzen. Auch die DOS-Meldungen und -Befehle lassen sich beispielsweise so verfremden.

Beide Modi (Hex/ASCII) werden durch Betätigen der Return-Taste beendet, wobei die Eingaben in der Anzeige auch in der jeweils anderen Darstellungsart

```
CSW = $36
ORG $6400
CALLRWTS LDA #>IOB
LDY #<IOB
JSR RWTS
BCC RET
RTS
RET LDA #0
STA IBSTAT
RTS
IOB
IBTYPE HEX 01
IBSLOT HEX 60
IBDRVN HEX 01
IBVOL HEX 00
IBTRK HEX 11
IBSECT HEX 0F
IBDCTP DA DCT
IBBUFF DA AREA
DUMMY DS 2
IBCMD HEX 01
IBSTAT HEX 00
IBSMOD HEX 00
IOBPSN HEX 60
IOBPDN HEX 01
```

```
DCT
DEVTPC HEX 00
PPTC HEX 01
MONTC HEX EF
HEX D8
```

```
RWTS EQU $3D9
YH HEX 00
PRINT HEX 00
SHOW LDY YH
TYA
JSR PRBYTE
LDA #": "
JSR OUT
LDA #$A0
JSR OUT
HEXD LDA AREA, Y
JSR PRBYTE
LDA #$A0
JSR OUT
INY
TYA
AND #$07
BNE HEXD
LDY YH
ASC LDA AREA, Y
CMP #$80
BMI LA1
CMP #$A0
BPL LA1
ORA #$40
LA1 JSR OUT
INY
TYA
AND #$07
BNE ASC
TYA
```

```
AND #$7F
BEQ END1
JSR CROUT
STY YH
JMP SHOW
END1 STY YH
RTS
OUT BIT PRINT
BMI SCREEN
CMP #$00
BNE NONULL
LDA #$A0
JMP SCREEN
NONULL CMP #$FF
BNE NOQ
LDA #$BF
JMP SCREEN
NOQ CMP #$A0
BCS SCREEN
CMP #$80
BCC INVFLS
CNTRL ORA #$40
JMP SCREEN
INVFLS AND #$3F
EOR #$20
CLC
ADC #$A0
SCREEN
JMP (CSW)
CROUT = $FD8E
PRBYTE = $FD8A
AREA = $6500
```

```
6400- A9 64 A0 10 20 D9 03 90 +=0349
6408- 01 60 A9 00 8D 1D 64 60 +=0278
6410- 01 60 01 00 11 0F 21 64 +=0107
6418- 00 65 36 B7 01 00 00 60 +=01B3
6420- 01 00 01 EF D8 00 00 AC +=0275
6428- 25 64 98 20 DA FD A9 BA +=047B
6430- 20 74 64 A9 A0 20 74 64 +=0339
6438- B9 00 65 20 DA FD A9 A0 +=045E
6440- 20 74 64 C8 98 29 07 D0 +=0358
6448- EF AC 25 64 B9 00 65 C9 +=040B
6450- 80 30 06 C9 A0 10 02 09 +=023A
6458- 40 20 74 64 C8 98 29 07 +=02C8
6460- D0 EA 98 29 7F F0 09 20 +=0413
6468- 8E FD 8C 25 64 4C 27 64 +=0377
6470- 8C 25 64 60 2C 26 64 30 +=025B
6478- 26 C9 00 D0 05 A9 A0 4C +=0359
6480- 9F 64 C9 FF D0 05 A9 BF +=0508
6488- 4C 9F 64 C9 A0 B0 10 C9 +=0441
6490- 80 90 05 09 40 4C 9F 64 +=02AD
6498- 29 3F 49 20 18 69 A0 6C +=025E
64A0- 36 00 +=0036
```

Bild 1. Source- und Objektlisting der Routinen CALLRWTS und SHOW. Sie müssen mit BSAVE DISK-EDITOR.OBJ, A\$6400, L\$A2 abgespeichert werden. Eine Eingabekontrolle ist mit dem Prüfsummen-Programm aus mc 1984, Heft 6, Seite 64, möglich

übernommen werden. Der so geänderte Sektorinhalt läßt sich nun mit dem Befehl SAVE auf die Diskette zurückschreiben. Dabei sollten Sie sich vorher vergewissern, daß Sie wirklich diese Änderungen gewünscht haben, da der Originalinhalt des Sektors nunmehr unwiderruflich überschrieben wird.

Über NEW können die Parameter neu gesetzt werden, um ein anderes Laufwerk oder einen Sektor am „anderen Ende“ der Diskette zu bearbeiten.

Tritt während des Betriebs ein Fehler auf, etwa aufgrund eines defekten Sektors oder weil die Diskette durch ein Spezialformat kopiergeschützt ist oder sein soll, so wird dies durch eine entsprechende Meldung angezeigt. In diesem Fall ist der angezeigte Sektorinhalt wertlos.

Das Programm selbst besteht aus zwei Teilen. Da ist zunächst der maschinensprachliche Teil (Bild 1) mit den Routinen CALLRWTS und SHOW. Er belegt den Speicher ab \$6400. Die Routine CALLRWTS ruft die RWTS (read/write track/sector) auf; sie verwendet dabei die unter IOB stehende Parameterliste. Für weitere Informationen über den Gebrauch sei auf [1], [2] und [3] verwiesen.

Die Routine SHOW stellt die ab AREA + YH abgelegten Bytes auf dem Bildschirm dar bzw. gibt sie auf den Drucker aus. Dabei ist YH = 0 für den ersten Halbsektor, YH = 128 für den zweiten. Kann Ihr Drucker invers drucken, so können Sie die Routine OUT sinngemäß verändern. In der abgedruckten Form werden alle Nichtstandardzeichen in solche umgeformt.

Bild 2 zeigt das Basic-Hauptprogramm. Dieses übernimmt Parametereingabe, Überschreiben des Sektorinhalts und Steuerung der Maschinenroutinen. Soll der Disk-Editor nur für 35-Track-Disketten verwendet werden, sollten Sie in 290 die Zahl 40 in 35 sowie in 300 die 39 in 34 umändern. Auch bei Verwendung von Original-Apple-Laufwerken ist dies angeraten, da diese nur 35 Spuren verarbeiten können. Benötigt Ihr Drucker besondere Sequenzen zum Ein- oder Ausschalten, so sollten Sie diese im 3000er-Bereich in die entsprechenden Zeilen einfügen.

So „repariert“ man Disketten

Im folgenden einige Tips zum Gebrauch des Disk-Editors zum Reparieren von

Disketten. Haben Sie ein File versehentlich gelöscht, danach aber keine weiteren Schreiboperationen auf der Disk vorgenommen, so kann es 100 %ig restauriert werden! Auf Track \$11 (17 dez.) befindet sich auf den Sektoren \$F (15 dez.) bis \$1 der Catalog. Suchen Sie dort mit Hilfe des Disk-Editors den Sektor, in dem der Name des gelöschten Files steht. 3 Bytes vor dem ersten Zeichen des File-Namens sollte das Byte \$FF stehen. Ersetzen Sie dieses durch das Byte, welches 32 Zeichen weiter unten steht. Dies finden Sie auch ohne Abzählen, indem Sie von \$FF aus vier Zeilen abwärts gehen. Nachdem Sie diese Änderung mittels SAVE auf der Diskette fixiert haben, ist das File wieder voll verfügbar.

Aus dem Catalog-Eintrag läßt sich auch die Verteilung einer Datei auf der Disk ermitteln. Das dritte Byte vor dem Filenamen enthält die Spur, das zweite den Sektor der „Track/Sektor List“ (TSL). In dieser TSL stehen ab Byte \$C fortlaufend die Track-/Sektor-Angaben für jeden Sektor des Files. Eine typische TSL zeigt Bild 3. Das zugehörige File belegt Track \$13, Sektor \$D bis \$1.

Sollten Sie also beim Laden oder Abspeichern eines Files einen I/O-Error erhalten, können Sie den defekten Sektor finden, indem Sie die in der TSL ange-

führten Sektoren der Reihe nach einzulesen versuchen. Beim Versuch, den defekten Sektor zu lesen, wird sich der Disk-Editor lautstark bemerkbar machen. Haben Sie so den defekten Sektor gefunden, entfernen Sie den zugehörigen Eintrag in der TSL und rücken mit den anderen auf. Sie können das File jetzt wieder einlesen und versuchen, den fehlenden Teil zu restaurieren.

Wenn der „Catalog“ kaputt ist...

Ein seltenes, nichtsdestotrotz hochnotpeinliches Ereignis ist der Verlust eines ganzen Catalog-Sektors oder sogar der ganzen Spur \$11. Selbst in diesem Fall ist nicht alles verloren; mit etwas mehr Arbeit läßt sich die ganze Diskette noch retten. Die Methode sei hier nur kurz skizziert. Für zusätzlich benötigte Informationen sei auf [1] verwiesen.

Ist Spur \$11 komplett hinüber, was beispielsweise passieren kann, wenn die Reset-Taste während eines Save-Vorgangs betätigt wird, so sind auf der Diskette immer noch von allen Files die TSLs vorhanden. Dank deren Aufbau sind sie auch bei schnellem Durchmusteren der Diskette noch zu erkennen. Mit dem Disk-Editor gehen Sie die ganze Diskette durch und notieren sich die Positionen aller „TSL-verdächtigen“ Sektoren. Haben Sie dann alle wahrscheinli-

```
100REM      DISK-EDITOR (C) MC / H.B.BIELING 1984
110GOSUB820
120GOSUB990
130GOSUB1180:CALLRWTS:VTAB21:HTAB13:E=PEEK(ERR)
140IFE=64THENPRINTBELL$;:FLASH:PRINT"DRIVE ERROR":NORMAL
150IFE=128THENPRINTBELL$;:FLASH:PRINT"READ ERROR":NORMAL
160IFE=0THENHTAB1:PRINTSPC(40)
170GOSUB1260
180HTAB1:VTAB24:PRINT"<E>DIT <S>AVE <N>EW <Q>UIT <C>OPY <- ->";
190GETA$:A=ASC(A$)
200IFA$="E"THEN350
210IFA$="S"THENGOSUB1330
220IFA$="N"THENRUN
230IFA$="Q"THENNEW
235IFA$="C"THEN3000
240IFA=8THENDIR=-1:GOTO270
250IFA=21THENDIR=1:GOTO270
260GOTO180
270IFDIR=1ANDPEEK(YH)=128THEN170
280IFDIR=-1ANDPEEK(YH)=0THEN170
290SEC=SEC+DIR
300IFSEC=16THENSEC=0:T=T+1
310IFSEC=-1THENSEC=15:T=T-1
320IFT=40THENT=0
330IFT=-1THENT=39
340GOTO130
350HTAB1:VTAB24:PRINT"<H>EX / <A>SCII"SPC(24);
360GETA$:IFA$="H"THEN630
370IFA$<>"A"THEN350
380P=0:HP=29:VP=4:VTABVP:HTABHP
```

Bild 2. Das Basic-Programm übernimmt die Steuerung der Maschinenroutinen und die Auswertung von Eingaben

```

390TY=1
400HTAB1:VTAB24:PRINT"ASCII ";TYP$(TY);" ";
410VTABVP:HTABHP
420GETA$:A=ASC(A$)
430IFA=27THEN470
440IFA=13THENPOKEYH,128-PEEK(YH):GOTO170
450GOSUB1440
460VTABVP:HTABHP:GOTO420
470TY=TY+1:IFTY=7THEN390
480HTAB1:VTAB24:PRINT"ASCII ";TYP$(TY);" ";
490HTABHP:VTABVP
500GETA$:A=ASC(A$)
510IFA=27THEN470
520IFA=13THENPOKEYH,128-PEEK(YH):GOTO170
530A=A+128
540ONTYGOSUBS550,550,560,570,580,590:GOTO600
550RETURN
560A=A+32:RETURN
570A=A+128-(A>=192)*64:RETURN
580A=A-64-(A>=192)*64:RETURN
590A=A-64:RETURN
600IFA<MIN(TY)ORA>MAX(TY)THEN490
610POKEPEEK(40)+PEEK(41)*256+PEEK(36),A
620POKEAREA+P+128-PEEK(YH),A:A$="K":GOSUB1460:GOTO490
630HTAB1:VTAB24:PRINT"HEX"SPC(33);
640P=0:HP=5:VP=4:VTABVP:HTABHP
650GETA$:IFA$>="0"ANDAS<="F"THEN760
660IFA$=CHR$(13)THEN810
670IFA="I"THENVP=VP-1:P=P-8
680IFA$="J"THENHP=HP-3:P=P-1
690IFA$="K"THENHP=HP+3:P=P+1
700IFA$="M"THENVP=VP+1:P=P+8
710IFHP=29THENHP=5:VP=VP+1
720IFHP=29THENHP=26:VP=VP-1
730IFVP=20THENVP=4:P=0
740IFVP=3THENVP=19:P=127
750VTABVP:HTABHP:GOTO650
760PRINTA$;A=ASC(A$)-48:IFA>9THENA=A-7
770GETA$:J=ASC(A$)-48:IFJ>9THENJ=J-7
780PRINTA$;
790J=16*A+J:IFJ<0ORJ>255THEN750
800POKEAREA+P-PEEK(YH)+128,J:A$="K":GOTO690
810POKEYH,128-PEEK(YH):GOTO170
820REM INITIALISIERUNG
830TEXT:HIMEM:100*256
840DINTYP$(6):FORI=1TO6:READTYP$(I):NEXT
850DATA CURSOR,NORMAL,KLEIN,INVERS,FLASH,CONTROL
860DIMHEX$(16)
870FORI=0TO15:READHEX$(I):NEXT
880DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
890DIMMIN(6),MAX(6):FORI=1TO6:READMIN(I),MAX(I):NEXT
900DATA 0,0,160,223,224,255,0,63,64,127,128,159
910DDS=CHR$(4)
920IFPEEK(25600)<169THENPRINTDDS"BLOAD DISK-EDITOR.OBJ
930YH=25637:POKEYH,0
940RWTS=25600:SHOW=25639:PR=SHOW-1
941POKEPR,255
950AREA=101*256
960ERR=25629
970BELLS=CHR$(7)+CHR$(7)
980RETURN
990REM PARAMETEREINGABE
100HOME:PRINT"*** HB DISK-EDITOR ***"
1010REM PARAMETEREINGABE
1020VTAB7:PRINT"ALLE EINGABEN DEZIMAL
1030VTAB10:PRINT"SLOT.....:"
1040VTAB12:PRINT"DRIVE.....:"
1050VTAB14:PRINT"TRACK.....:"
1060VTAB16:PRINT"SECTOR.....:"
1070VTAB10:HTAB17:INPUTSL$:SL=VAL(SL$)
1080IFSL<1ORS�>7THEN1070
1090VTAB12:HTAB17:INPUTD$:D=VAL(D$)
1100IFD<1ORD>2THEN1090
1110VTAB14:HTAB17:INPUTT$:T=VAL(T$)
1120IFT<0ORT>40THEN1110
1130VTAB16:HTAB17:INPUTSECS$:SEC=VAL(SECS$)
1140IFSEC<0ORSEC>15THEN1130
1150PRINT:PRINT"OK (Y/N) ? ";GETA$:IFA$<>"Y"THEN1070
1160HOME
1170RETURN
1180REM SETUP RWTS
1190POKE25617,16*SL
1200POKE25618,D
1210POKE25619,0
1220POKE25620,T
1230POKE25621,SEC
1240POKE25628,1:REM DISK-LESEN
1250RETURN
1260REM DATEN AUF SCHIRM ANZEIGEN
1270VTAB3:HTAB7
1280T1=INT(T/16):T2=16*(T/16-T1)
1290S1=INT(SEC/16):S2=16*(SEC/16-D1)
1300PRINT"TRACK: ";HEX$(T1);HEX$(T2);" SECTOR: ";HEX$(S1);HEX$(S2)
1310CALLSHOW
1320RETURN
1330REM DISK-SCHREIBEN
1340POKE25628,2:REM DISK-SCHREIBEN
1350CALLRWTS:VTAB21:HTAB10:E=PEEK(ERR)
1360IFE<64THEN1380
1370PRINTBELLS;HTAB13:FLASH:PRINT"DRIVE ERROR":NORMAL
1380IFE<128THEN1400
1390PRINTBELLS;HTAB13:FLASH:PRINT"READ ERROR":NORMAL
1400IFE<16THEN1420
1410PRINTBELLS;HTAB10:FLASH:PRINT"WRITE PROTECTED":NORMAL
1420IFE=0THENHTAB1:PRINTSPC(40)
1430RETURN
1440IFA$="I"THENVP=VP-1:P=P-8
1450IFA$="J"THENHP=HP-1:P=P-1
1460IFA$="K"THENHP=HP+1:P=P+1
1470IFA$="M"THENVP=VP+1:P=P+8
1480IFHP=37THENHP=29:VP=VP+1
1490IFHP=28THENHP=36:VP=VP-1
1500IFVP=20THENVP=4:P=0
1510IFVP=3THENVP=19:P=127
1520RETURN
3000REM DRUCKER INITIALISIEREN
3100PR#1:PRINT:PRINT:PRINT
3110POKEPR,0
3115POKEYH,128-PEEK(YH)
3120GOSUB1260
3125REM DRUCKER AUS
3130PR#0:TEXT:POKEPR,255:GOTO180

```


chen TSLs zusammengetragen (wahrscheinlich deshalb, weil einige von ihnen von gelöschten und teilweise überschriebenen Files stammen können), beginnt die eigentliche Arbeit. Zunächst muß Spur \$11 wieder benutzbar gemacht werden.

mc-Leser werden sich des Artikels von Wolfgang Schöpe aus dem Aprilheft 1984 [4] erinnern, der das „Aufbohren“ von Apple-Disketten von 35 auf 40 Spuren beschrieb. Dazu mußten die Spuren 35 bis 39 gesondert formatiert werden. Indem man das damalige Basic-Hauptprogramm etwas zweckentfremdet, kann man es gezielt zum Erstellen einer neuen Spur \$11 verwenden. Bild 4 zeigt das neue Basic-Programm. Nach dessen Anwendung hat man einen leeren Catalog, den es mit Hilfe des Disk-Editors zu füllen gilt.

TRACK: 13 SECTOR: 0F

```
00: 00 00 00 00 00 00 00 00 00
08: 00 00 00 00 00 13 0D 13 0C SMSL
10: 13 0B 13 0A 13 09 13 08 SKSJISISH
18: 13 07 13 06 13 05 13 04 SGSFSESD
20: 13 03 13 02 13 01 00 00 SCBSBA
28: 00 00 00 00 00 00 00 00
30: 00 00 00 00 00 00 00 00
38: 00 00 00 00 00 00 00 00
40: 00 00 00 00 00 00 00 00
48: 00 00 00 00 00 00 00 00
50: 00 00 00 00 00 00 00 00
58: 00 00 00 00 00 00 00 00
60: 00 00 00 00 00 00 00 00
68: 00 00 00 00 00 00 00 00
70: 00 00 00 00 00 00 00 00
78: 00 00 00 00 00 00 00 00
```

Bild 3. So sieht eine typische Track-/Sector-Liste aus. Sie beschreibt ein File, welches auf Track \$13 die Sektoren \$D bis \$1 belegt

```
140BELL$=CHR$(7)+CHR$(7)
150PRINTCHR$(4)"BLOAD TRK.OBJ"
160HOME:VTAB5
170PRINT"DIESES PROGRAMM FORMATIERT"
171PRINT"TRACK 11 NEU. LEGEN SIE DIE"
172PRINT"ZU BEARBEITENDE DISKETTE EIN"
173PRINT"UND DRUECKEN SIE DIE LEER-"
174PRINT"TASTE. A C H T U N G : "
175PRINT"DER CATALOG DER DISK WIRD"
176PRINT"DABEI UEBERSCHRIEBEN! ";
220GETR$:POKE25,1:POKE27,0:POKE28,0
240POKE29,0:POKE26,17:CALL847
290IFPEEK(29)>0THEN330
320GOTO480
330PRINT:PRINTBELL$"SCHREIBGESCHUETZT!"
335FORI=1TO2000:NEXT:GOTO160
480POKE26,17:POKE27,0:POKE28,1:CALL768
500RESTORE
510FORI=4097TO4102
520READW:POKEI,W:NEXT
521FORI=4144TO4151:READW:POKEI,W:NEXT
550POKE4148,40:POKE28,2:CALL768
580PRINT:PRINT"REPARATUR BEENDET."
620DATA 17,15,3,0,0,255
630DATA 0,0,0,0,35,16,0,1
```

Bild 5. Das Maschinenprogramm ist nach Eingabe mit BSAVE TRK.OBJ, A\$300, L\$3C0 abzuspeichern

```
0300- 20 E3 03 84 06 85 07 A0 +=02BC
0308- 01 A9 60 91 06 C8 A5 19 +=0327
0310- 91 06 C8 A9 00 91 06 C8 +=0367
0318- A5 1A 91 06 C8 A5 1B 91 +=036F
0320- 06 A0 08 A9 00 91 06 C8 +=02B6
0328- A9 10 91 06 A0 0C A5 1C +=02BD
0330- 91 06 A0 0E A9 00 91 06 +=0285
0338- C8 A9 60 91 06 A4 06 A5 +=03B7
0340- 07 20 D9 03 A9 00 85 48 +=0279
0348- A0 0D B1 06 85 1D 60 20 +=0286
0350- E3 03 84 06 85 07 A0 01 +=029D
0358- A9 60 91 06 C8 A5 19 91 +=03B7
0360- 06 C8 A9 00 91 06 C8 A5 +=037B
0368- 1A 91 06 A0 0C 91 06 A4 +=0298
0370- 06 A5 07 20 D9 03 BD 89 +=02F4
0378- C0 A5 1A 85 44 A9 00 85 +=0376
0380- 41 A9 AA 85 3E A9 28 85 +=03AD
0388- 45 A9 60 8D CB BE 8D F7 +=04E8
0390- BE A0 56 A9 00 20 BF BE +=03FA
0398- 20 0D BF A9 08 B0 05 20 +=0272
03A0- E0 BE 90 06 A0 0D B1 06 +=0398
03A8- 85 1D A9 A9 8D CB BE A9 +=04B3
03B0- B0 8D F7 BE BD 88 C0 A9 +=05A0
03B8- 00 85 48 60 2C 8B C0 +=02A4
```

...ist noch nichts verloren

Ist „nur“ ein Catalog-Sektor ruiniert, kann man die Catalog-Verkettung um diesen Sektor herumbiegen und versuchen, die verschwundenen Files wiederzufinden. Anhand der gefundenen TSLs werden die Catalog-Einträge restauriert. Den genauen Aufbau eines Catalog-Sektors entnehmen Sie [1], oder Sie schauen sich den Catalog einer intakten Diskette als Beispiel an. Beim Wiederherstellen sollten Sie zunächst Namen wie T1, T2 usw. verwenden, bis Sie sich über die ursprüngliche Identität des Files im klaren sind. Zur Entscheidung, ob es sich um ein Basic-, Binär- oder Textfile handelt, sollten Sie sich den ersten in der TSL angeführten Sektor ansehen. Ein Textfile erkennt man sofort an der ASCII-Darstellung. Bei einem Basic-File

enthalten die beiden ersten Bytes die Länge des Programms, danach folgt das „tokenisierte“ Programm. Hat das vierte Byte den Wert \$08, so handelt es sich vermutlich um Basic. Der Typ des Files (0 = Text, 2 = Basic, 4 = Binär) wird im Catalog-Eintrag durch das erste Byte vor dem Filenamen festgelegt. Die Filelänge sollten Sie mit \$FF angeben, das reicht garantiert noch für die längsten Files.

Haben Ihre Nerven diese Arbeit (die bei vielen Files durchaus Stunden dauern kann) überstanden, können Sie jetzt darangehen, die Files T1, T2 usw. in gut und böse zu scheiden; gut die, die tatsächlich einen Sinn ergeben, böse die, die alte, teilweise überschriebene Rudimente von irgend etwas sind. Bei Basic-Programmen und Textfiles ist das relativ leicht zu entscheiden, bei Binärfiles hilft etwas Fingerspitzengefühl. Vielleicht probieren Sie auch das Verfahren einmal mit einer Sicherungskopie aus, damit Sie es beherrschen, sollten Sie es einmal brauchen. Und vielleicht bewahrheitet sich dann Murphy's Law, daß alles gutgeht, wenn man auf das Schlimmste gefaßt ist.

Bild 4. Zum gezielten Formatieren von Track \$11 dient dieses Programm in Verbindung mit dem Maschinenprogramm TRK.OBJ aus Bild 5. Vorsicht bei der Anwendung, da ein heiler Catalog restlos gelöscht wird

Literatur

- [1] Apple Computer Inc.: The DOS Manual.
- [2] Wiegandt, Dr. Ralf: Apple-DOS Arbeitsweise und Aufbau. mc 1983, Heft 6, Seite 53.
- [3] Worth/Lechner by QS: Beneath Apple DOS.
- [4] Schöpe, Wolfgang: Mehr Platz auf Apple-Disketten. mc 1984, Heft 4, Seite 72.

Nachträge

Universal-Schnittstelle für Apple-II

mc 1983, Heft 4, Seite 102
Die Verzögerungsschaltung für das Enable-Signal des VIA-Bausteins 6522 verzögert nicht, wie im Text beschrieben, die positive Flanke des E-Signals gegenüber der negativen Flanke des $\Phi 1$ -Signals, da

durch einen unglücklichen Zeichenfehler aus dem erforderlichen NOR-Gatter ein „ $\frac{1}{4}$ 74LS00“ wurde. Die richtige Schaltung zeigt *Bild 1*. Eine weitere Modifikation, die sich als günstig erwiesen hat, zeigt *Bild 2*.

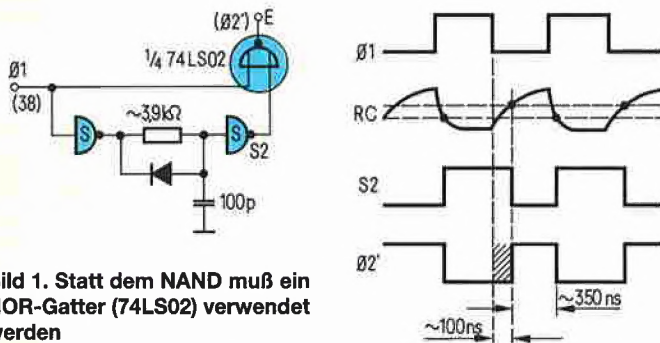


Bild 1. Statt dem NAND muß ein NOR-Gatter (74LS02) verwendet werden

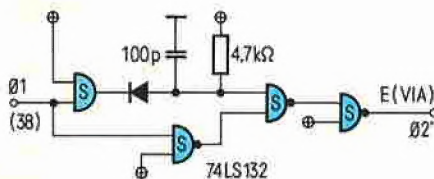


Bild 2. Hier wird zur Bildung des Signals E ein 74LS132 verwendet

Apple-Kniffe

mc 1983, Heft 12, Seite 53

Die Änderung des Basic-Prompts kann zur Folge haben, daß ab und zu DOS-Befehle nicht ausgeführt werden. Die Ursache für den „Syntax Error“ ist, daß DOS 3.3 das Prompt-Zeichen auf hex DD abfragt (Adresse hex A66C bei min. 48 KByte RAM). Abhilfe schafft der Befehl POKE 42604, PEEK (54336). Er installiert das eventuell geänderte Prompt-Zeichen auch im DOS. Sinnvollerweise schreibt man diese Anweisung an den Anfang des HELLO-Programms, das beim Booten immer zuerst ausgeführt wird.

Die von unserem Software-Service gelieferten EPROMs (D000...D7FF) enthalten eine weitere nützliche Änderung: Durch hex 3F statt 3C bei D50D entstehen bei der Eingabe von Basic-Programmen keine unnötigen Leerzeilen mehr. Der Franzis-Software-Service liefert auf der Apple-Sammeldiskette 3 (18,50 DM) auch ein Execute-File, das die geänderte Basic-Version in den RAM-Bereich ab D000 schreibt (64 KByte RAM bzw. 16-KByte-Language-Karte erforderlich). Außerdem enthält diese neue Diskette das an den Apple-II angepaßte Basic-calc-Programm aus Heft 12/1983 und zahlreiche andere nützliche Routinen.

Apple-II lernt sprechen

mc 1983, Heft 6, Seite 81

Leider wurde ein Teil des Hex-Dumps nicht mit abgedruckt, was beim Vergleich mit dem (vollständigen) Assemblerlisting gleich auffällt. Das *Bild* zeigt das fehlende Stück. Bei Verwendung als Unterprogramm kann der Inhalt der Speicherzelle hex 0943 auf hex 60 geändert werden.

87E.883

087E- A6 FF
0880- CA D0 FD 60

* 900.945

0900- A9 01 85 F9 A2 00 A9 00
0908- 85 FB A5 FD 85 FC A1 FB
0910- 25 F9 C5 FA 85 FA F0 06
0918- 2C 30 C0 4C 23 09 EA EA
0920- 4C 23 09 20 7E 08 18 A5
0928- FB 69 01 85 FB A5 FC 69
0930- 00 85 FC C5 FE D0 07 A5
0938- F9 C9 80 F0 06 0A 85 F9
0940- 4C 0A 09 4C 69 FF
*

Die zweite Änderung ist am einfachsten, wenn man das Programm neu assemblieren kann. Es muß nämlich ein zusätzlicher LSR-Befehl vor (!) folgender Adresse eingefügt werden:

Apple-II-Komm. 036F
Apple-Modempgm. 09C8
Apple-Modemp/FSS 85C8
Beim MC-65-Programm geht es einfacher: Hier muß nur der AND-Befehl bei 0340 (29 7F) durch die Befehlsfolge LSR, NOP (4A EA) ersetzt werden. Bei den anderen drei Programmversionen muß man, falls kein Assembler zur Verfügung steht, einen kleinen „Patch“ an einer freien Adresse einbauen:

Apple-II-Kommunikationsprogramm:

036D: 4C 24 03
0324: A5 FD 4A 09 80 30 46
Apple-Modemprogramm:
09C6: 4C B4 0B
0BB3: 00 A5 FB 4A 84 F7 4C
CA 09

Apple-Modemprogramm, FSS-Version:

85C6: 4C B4 87
87B3: 00 A5 FB 4A 84 F7 4C
CA 85

Da sich bei keinem der Programme seine Länge ändert, ist das Abspeichern wie bisher möglich. Der Kontakt mit unserem Teledaten-Service TEDAS wird übrigens durch die Umstellung auf ein Stopbit nicht beeinträchtigt.

Modem-Programme für 6502-Computer

mc 1/1984, mc 3/1984 und mc-Modem-Sonderheft

Das Apple-Kommunikationsprogramm aus mc 1/1984, das MC-65-Modemprogramm (mc 3/1984 und Modem-Sonderheft) sowie das Apple-II-Modemprogramm aus dem Modem-Sonderheft empfangen die Zeichen mit einer Interrupt-Routine, die für zwei Stopbits ausgelegt ist. Für den Kontakt mit Datex-P20F (300 Bd) ist aber ein Stopbit nötig. Dazu muß man zunächst den Bitzähler der Interrupt-Routine von 09 auf 08 ändern. Seine Adresse ist: Apple-II-Komm. 0359
MC-65-Modempgm. 032A
Apple-Modempgm. 09B2
Apple-Modemp/FSS 85B2
(Der Zusatz FSS = Franzis-Software-Service bezieht sich auf das in den Adressenbereich ab 8400 verschobene Modem-Programm auf der Apple-Sammeldiskette 5.)

Software-Service

Zahlreiche Programme aus diesem Sonderheft können Sie von unserem Software-Service auch auf Diskette beziehen, um sich ein zeitaufwendiges und fehlerträchtiges Eintippen zu ersparen. Bitte fordern Sie eine vollständige Liste unserer Apple-Disketten an bei: Franzis-Software-Service, Postfach 37 01 20, 8000 München 37, Tel. 0 89/ 51 17 3-31. Jede Diskette kostet nur 20 DM.

Shapemaker

Das in mc 1983, Heft 3, veröffentlichte Programm „Shapemaker“ für den Apple-II beinhaltet ein Problem: In ein Byte können bis zu drei Vektoren „gepackt“ werden. Wenn aber der zweite Vektor „Kein Punkt, nach oben“ lautet (SU entspr. 000) und der dritte Vektor ein zu zeichnender Vektor ist (PL, PR, PU, PD) oder wieder SU, dann muß dieses Byte schon nach dem ersten Vektor beendet werden, und der zweite und dritte Vektor müssen in das nächste Byte gepackt werden. Nachfolgend eine Lösung des Problems. Da die neue Programmzeile zwischen die Zeilen 1096 und 1097 geschoben oder an die Zeile 1096 angehängt werden muß, habe ich sie 1096a genannt:

```
1096a IF J=0 AND IN$
(J+1)=„SU“ AND
(IN$(J+2)=„PL“ OR
IN$(J+2)=„PR“ OR
IN$(J+2)=„PU“ OR
IN$(J+2)=„PD“ OR IN$
(J+2)=„SU“) THEN
DX=0:J1=2
```

Der Spezialfall wurde vom Autor nicht beachtet, so daß es ohne diese Zeile zu unerklärlichen Verstümmelungen der Shapes kommt.

*Michael Ritter,
Pinneberg*

Bei dem Programm Shapemaker für den Apple-II (mc 1983, Heft 3, Seite 74) zeigte sich leider, daß in einigen Fällen Shapes nicht richtig umgesetzt wurden. In einer neuen Programmversion sind diese Fehler behoben. Außerdem wurde es in den folgenden Punkten komfortabler:

- Die Eingabe der Shape-Länge entfällt.
- Es werden keine Bytes mehr verwendet, da die Shapes nun unmittelbar hintereinander abgespeichert werden.
- Die Anfangsadresse der

Shape-Tabelle kann allein in Zeile 1002 geändert werden.

- Die maximale Shape-Länge kann in Zeile 1003 geändert werden.
- Man kann in der Shapetabelle „blättern“ und ab einem früheren Shape überschreiben.
- Die Shapetabelle kann auch dann abgespeichert werden, wenn nicht alle Shapes eingegeben wurden.
- Das Abspeichern auf Kassette wurde komfortabler

*Heinrich Neupert,
Nürnberg*

*Das verbesserte Programm kann auf Diskette vom Franzis-Software-Service gegen eine Schutzgebühr bezogen werden.
Die Red.*

Step und Trace für Apple-II

Der Autor des Beitrags in mc 1/1984, Seite 91, hat leider drei Segmente des Apple-Monitors nicht mit in die Language-Karte übernommen. Punkt 2 der Eingabe muß also folgendermaßen geändert werden:

```
FA40...FB1D übertragen nach 8340...841D
FB60...FBC0 übertragen nach 8460...84C0
```

Somit wird der Bereich FAD7...FAFC zusätzlich eingefügt. *Dirk Ottensmeyer,
Hannover*

Apple-Eigenheiten

Sie schrieben in Ihrem Beitrag „Apple-Eigenheiten“ (mc 12/1983), daß eine Parameter-Übergabe mit dem CALL-Befehl nicht möglich sei. Mit einem Trick geht es doch: Es können beliebig viele Parameter übergeben werden, wenn man in die Adresse, zu der der CALL-Befehl springt, JSR

\$E6F5 schreibt. Das im Apple-soft-Interpreter dort stehende Unterprogramm holt eine numerische Variable oder eine Zahl aus dem Basic-Programm, die allerdings nicht kleiner als Null, nicht größer als 255 und ganzzahlig sein muß. Die Variable oder Zahl wird mit einem Komma von der CALL-Adresse getrennt:

```
CALL adr, numvar1,
numvar2...
```

Das Unterprogramm kehrt mit dem Wert der Variablen oder der Zahl im X-Register zurück. Man kann diesen Wert jetzt irgendwo im Speicher mit STX retten und die nächste Variable aus dem Basic-Programm holen. *Thomas Schweikle,
Schömberg-Schörzingen*

Apple-Grafik füllt eine DIN-A4-Seite

Das in mc 2/1984 auf Seite 66 abgedruckte Assembler-Programm für den MX-82 ist zwar, wie im Beitrag erwähnt, auch mit dem Drucker MX-100 lauffähig, bringt das Apple-Schirmbild aber nur unvollständig auf Papier. Der MX-100 kann bei einfacher Druckdichte horizontal 816 Punkte pro Zeile drucken, der MX-82 aber nur 576. Damit die geraden Zeilennummern auch unter den ungeraden stehen und nicht teilweise rechts davon, muß nach 576 Punkten ein „Carriage Return“ veranlaßt werden. Eine kleine Änderung im Assembler-Programm bewirkt das. Hinter Zeile 57 muß eingefügt werden: JSR CRLF.

Da sich das Programm von dort an um drei Bytes verschiebt, muß auch das Basic-Programm geändert werden:

```
350 IF Z=1 THEN POKE
829,6:POKE 920,76:POKE
925,128:POKE 930,4
370 IF Z=3 THEN POKE
883,64
```

*Hartmut E. Püchner,
Frankfurt*

Apple druckt Grafik

Ich freue mich jedesmal auf ihre ausgezeichnete Zeitschrift. In der Ausgabe vom Februar 1984 erschien der Artikel „Apple-Grafik füllt eine DIN-A4-Seite“ von Wolfgang Ebner. Das Programm ist gut und läuft auf dem Apple-IIe. In Verbindung mit dem Epson-Drucker FX-80 werden aber von den 192 Punkten nur 160 ausgedruckt. Geht man dem Problem auf den Grund, wird ersichtlich, daß nur 480 Druckpunkte ausgegeben werden (wie beim MX-80). Erforderlich sind, wie im Artikel dargelegt, 576 Druckpunkte. Ein Blick in das FX-80-Handbuch auf Seite 102 zeigt, daß der FX-80 auch 576 Druckpunkte ausgeben kann, und zwar mit dem Befehl:

```
:CHR$(27);„ö“;CHR$(5);CHR-
$(64);CHR$(2);
```

Im Programm von Herrn Ebner wird in der PINIT-Routine der Befehl :CHR\$(27);„K“;... aus der Seite 130 im FX-80-Handbuch verwendet. Man braucht also lediglich die PINIT-Routine entsprechend zu ändern, und die ganze Seite erscheint auf dem FX-80.

Im einzelnen sieht das so aus: Adresse 395 von 4B zu 2A ändern, Adressen 399 bis 3CD um fünf Bytes verschieben und in die Lücke bei Adresse 399 die Bytes A9 05 20 86 03 eingeben.

Durch die Verschiebung ändern sich die Anfangsadressen HSCRN von 3A4 auf 3A9, DECXL von 3B8 auf 3BD und CRLF von 3C3 auf 3CB. Diese sind entsprechend zu ändern. Im Basic-Programm lautet die Zeile 350 neu:

```
IF Z = 1 THEN POKE 829,5:
POKE 922,1:POKE 927,192:
POKE 932,3
```

In der doppelten Druckdichte werden fünf Druckpunkte für einen Bildpunkt gesetzt.

*Heinrich Suter,
Suhr/Schweiz*



Die Mikrocomputer-Zeitschrift, die ihre Leser zu Profis macht:

MC liefert Grundlagen für alle, die sich mehr als nur vordergründig mit der Mikrocomputerei befassen möchten...

MC informiert umfassend. Über Computer und Peripherie, über Programmiersprachen und Betriebssysteme...

MC regt an, auch mal etwas selbst zu bauen. Denn MC präsentiert Applikationen vom einfachen Interface bis zum kompletten Selbstbausystem.

MC kann man ganz einfach kennenlernen. Die nebenstehende Kennenlernkarte ist dafür bestimmt.

MC setzt allgemeines technisches Verständnis voraus, weil sie den ernsthaft Interessierten weiterbringen will...

MC testet Hardware und prüft Programme. MC gibt so Entscheidungshilfe vor einer Anschaffung.

MC hat auf alle Fragen zur Computertechnik eine Antwort. Mit Hilfe Ihres Computers und eines Telefonmodems können Sie Programme und Literaturstellen direkt bei MC abrufen...



Die Mikrocomputer-Zeitschrift



Berlin



Hamburg

- Runow Büroelektronik –
Ihr Apple-Vertragshändler
mit
- * Service – Level 1
 - * Fachberatung – Level 1
 - * großem Zubehörprogramm
 - * Computererfahrung seit mehr als 10 Jahren
 - * eigener Service- und Wartungswerkstatt



Mac Type für Macintosh

Die Mac Type-Typenradschreibmaschine ist eine ideale Ergänzung für Ihr Computersystem und direkt anschließbar:

- * Brother CE 50 - Mac
- * Brother CE 60 - Mac
- * Brother CE 61 - Mac

Andere Schnittstellen:
CBM-IEC · CBM-C64 · HP-IL · Centronics

Infos: PAC-Hardware GmbH
Keithstraße 26 · 1000 Berlin 30
Telefon 0 30-26 111 26

Runow Büroelektronik

1 Berlin 30 · Keithstr. 26 · 0 30-26 111 26
2 Hamburg 76 · Bachstr. 104 · 0 40-220 11 55



**Autorisierter apple-Händler
& Service-Center Level 1**

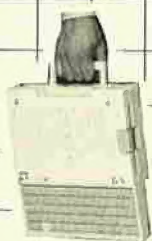
Macintosh
von Apple

Einfach.
Vielseitig.



Der Apple IIc

hat's in sich.



128 KB Arbeitsspeicher, eingebautes Diskettenlaufwerk (143 KB), 80/40 x 24 Zeichen Darstellung, Anschlüsse für: Maus, Joystick und Handregler, Modem, Fernsehgerät und RGB-Farbmonitor, flachen LC-Bildschirm, externes Diskettenlaufwerk, Drucker und Farbplotter. Apple II kompatibel

Drucker und Zubehör: NEC, Olympia, Epson, ITOH, CompuMate, Sanyo, Zenith, Apple.
Informationsmaterial gegen DM 3,- Rückporto (Briefmarken)

apple IIc apple III Lisa

BROSIOUS-KÖHLER GmbH & Co KG, Computersysteme

Unterdomen 71
5600 Wuppertal 2
Tel.: (02 02) 55 40 55-56
Telex 8 592 042 buk d
Mo.-Fr. 9-18 Uhr

Wuppertal 16
5650 Solingen 1
Tel.: (02122) 10038-39
Mo.-Fr. 9-18.30 Uhr
Sa. 9-13 Uhr



MiCOM
Computersysteme

64 KByte RAM
14 KByte ROM
650 Z + Z 80 A (ZMHZ)
Doppelprozessorsystem
Voll Apple-Kompatibel

- 1/2 Jahr Garantie
- CP/M & PASCAL - fähig
- 100% getestet: absolut problemlos trotzdem preiswert
- in verschiedenen Ausstattungen und Gehäusen erhältlich
- Floppies, Erweiterungsplatinen, Monitore, Disketten, Zubehör
- Beratung & Service
- Bitte fordern Sie unseren kostenlosen Katalog an.

MiCom-Computer Olaf Mertens

Industriehof Lüttringhausen - Grünenplatzstr. 16-18
5630 Remscheid 11
Tel. 02191/590313 - Telex 8513924 indu d